UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS

FACULTAD DE INGENIERÍA DE SISTEMAS E INFORMÁTICA

UNIDAD DE POSGRADO

"HERRAMIENTA PARA EL MODELADO DE LA REPLICACIÓN DE MYSQL BASADA EN LA INGENIERÍA DIRIGIDA POR MODELOS"

TESIS

Para optar el grado académico de Magíster en Ingeniería de Sistemas e Informática con Mención en Ingeniería de Software

AUTOR

Efraín Ricardo Bautista Ubillús

ASESOR

Nora La Serna Palomino

Lima – Perú

2014

Efraín Ricardo Bautista Ubillús

Herramienta para el Modelado de la Replicación de MySQL Basada en la Ingeniería Dirigida por Modelos

"Tesis presentada a la Universidad Nacional Mayor de San Marcos, Lima, Perú, para obtener el grado de Magíster en Ingeniería de Sistemas, en la mención de Ingeniería de Software"

Orientadora: Dra. Nora La Serna

UNMSM – LIMA

Mayo 2014

FICHA CATALOGRÁFICA

HERRAMIENTA PARA EL MODELADO DE LA REPLICACIÓN DE MYSQL BASADA EN LA INGENIERÍA DIRIGIDA POR MODELOS

Efraín Ricardo Bautista Ubillús
Lima – Perú, 2014
Orientadora: Dra. Nora La Serna Palomino
Disertación: Magíster en Ingeniería de Sistemas e Informática.

Universidad Nacional Mayor de San Marcos
Escuela de Posgrado
Facultad de Ingeniería de Sistemas e Informática, 2014
Unidad de Posgrado.
Páginas: 138.

Este trabajo está dedicado a toda mi familia, en especial a mi amado y adorado hijo Thiago Ricardo.

AGRADECIMIENTOS

A mi asesora Dra. Nora La Serna Palomino, por su constante apoyo, orientación, dedicación y revisiones para que este trabajo cumpla con los objetivos trazados, y porque en todo momento me animó para que culmine esta investigación.

Al profesor Dr. David Mauricio Sánchez por su orientación, consejos y revisiones del presente trabajo.

A mi hijo Thiago Ricardo, por ser mi mayor fuente de motivación.

A mi esposa Lucero, por su amor, paciencia, comprensión y apoyo incondicional para poder culminar este trabajo.

A mis cuatro padres, Lucio, Esther; Hugo y Virginia, a quienes debo mi educación y formación, y por todo su amor.

A todos mis profesores de maestría de la UNMSM, por compartir sus conocimientos que me sirvieron de mucho en este trabajo.

A mis amigos y colegas en Surgeon's Advisor de Miami y SoftBrilliance en Lima, por el apoyo brindado.

A todas aquellas personas que indirectamente me ayudaron para culminar este trabajo y que muchas veces constituyen un invalorable apoyo.

Y por encima de todo doy gracias a Dios.

Herramienta para el Modelado de la Replicación de MySQL Basada en la Ingeniería Dirigida por Modelos

RESUMEN

Para modelar la replicación de MySQL, los administradores de base de datos (DBA's) utilizan herramientas de diagramación, tales como Microsoft Visio. Sin embargo, este tipo de herramientas no permiten validar automáticamente si un modelo de replicación MySQL está libre de errores, lo que puede resultar tener documentación errónea de los modelos de replicación. Estas herramientas tampoco permiten generar automáticamente a partir del modelo, los comandos mysqlreplicate de configuración. La falta de estas funcionalidades conlleva a realizarlas de forma manual, la cual se torna en una tarea tediosa, propensa a errores y que consume tiempo, más aún si el número de servidores MySQL del modelo es alto, con 15, 20, 25 o más servidores.

Este trabajo de investigación propone el desarrollo de la herramienta MySQL Replication Modeling que permita a los DBA's modelar la replicación de MySQL y validar automáticamente si el modelo es correcto, mostrando los errores en caso existan. Además, una vez que el DBA ha corregido y validado el modelo, la herramienta es capaz de generar automáticamente los comandos mysqlreplicate de configuración. La herramienta se desarrolló siguiendo las fases del Proceso Unificado de Rational (RUP) y basada en la Ingeniería Dirigida por Modelos (MDE) bajo la plataforma Eclipse.

Los resultados demuestran que la herramienta propuesta MySQL Replication Modeling permite validar automáticamente un modelo de replicación de MySQL, y reduce en más del 87% el tiempo en identificar y corregir los errores de un modelo de replicación con 25 servidores comparado con el uso de la herramienta Microsoft Visio 2013. Los resultados también demuestran que con el uso de la herramienta propuesta se reduce en más del 99% el tiempo para generar los comandos mysqlreplicate de configuración comparado con el uso de la herramienta Microsoft Visio 2013.

Palabras Clave: Modelado de la Replicación de MySQL, Ingeniería Dirigida por Modelos, MDE.

MySQL Replication Modeling Tool Based on Model Driven Engineering

ABSTRACT

To model the MySQL replication, the database administrators (DBAs) use diagramming tools, such as Microsoft Visio. However, this type of tools do not allow automatically validating if the MySQL replication model is free of errors. Thus, we can have erroneous documentation of the MySQL replication models. These tools also do not allow automatically generating from the model, the mysqlreplicate commands of configuration. Due to the lack of these features, these are done manually, which becomes a tedious task, error prone, and time consuming, especially if the number of servers MySQL of the model is high, with 15, 20, 25 or more servers.

This research proposes the development of the tool MySQL Replication Modeling that allows to the DBA's modeling the MySQL replication and automatically validate if the model is correct, showing if there are errors. In addition, once the DBA has fixed and validated the model, the tool is capable of generating the mysqlreplicate commands of configuration. The tool was developed following the phases of the Rational Unified Process (RUP) and based on the Model Driven Engineering (MDE) under the Eclipse platform.

The results demonstrate that the proposed tool MySQL Replication Modeling allows automatically validate a MySQL replication model, and reduce in more than 87% the time to identify and fix a MySQL replication model with 25 servers compared with the use of the tool Microsoft Visio 2013. The results also demonstrate that with the use of the proposed tool the time to generate the mysqlreplicate commands of configuration is reduced in more than 99% compared with the use of the tool Microsoft Visio 2013.

Keywords: MySQL Replication Modeling, Model Driven Engineering, MDE.

ÍNDICE

Lista de Figuras

Lista de Tablas

Capítulo 1: Introducción

1.1 Antecedentes

1.1.1 Antecedentes del Problema

MySQL es el sistema de gestión de base de datos relacional (RDBMS) de software libre (open source) más popular, exitoso y usado del mundo [Ahmed+ 2010], [Li+ 2013], [Mahanta+ 2013] y [MySQL 2013a].

MySQL soporta nativamente la replicación de base de datos, esta técnica es ampliamente usada por los administradores de base de datos (DBA's) y profesionales en MySQL por ser la estrategia más común y económica para mejorar el rendimiento, escalabilidad y disponibilidad en las aplicaciones basadas en MySQL [MySQL 2013e] y [MySQL 2013g].

Existe una gran variedad de herramientas diseñadas exclusivamente para la configuración, administración y monitoreo de la replicación de MySQL, tales como MySQL Utilities [MySQL 2013f], Percona Toolkit [Percona 2011], MySQL Enterprise Monitor [MySQL 2013h], etc.

Sin embargo, no existe alguna herramienta diseñada exclusivamente para modelar la replicación de MySQL, motivo por el cual los DBA's utilizan herramientas de diagramación de propósito general tales como Microsoft Visio [Visio 2013a], Dia [Dia 2013], draw.io [Draw.io 2013], etc.

Un DBA por más experimentado que sea puede cometer algún error al diagramar las conexiones de su modelo de replicación de MySQL utilizando cualquier herramienta de diagramación, incrementándose más la probabilidad de cometer errores si el número de replicación de servidores del modelo es alto, como consecuencia se tiene documentación errónea de los modelos de replicación.

La validación mediante la identificación de errores de un modelo de replicación es una tarea manual realizada por el DBA, así como la escritura de los comandos mysqlreplicate para configurar un modelo de replicación.

Estas tareas manuales son tediosas, propensas a errores, y que consume tiempo que puede ser empleado en otras actividades críticas propias de un DBA.

1.1.2 Antecedentes de la Técnica

Desarrollar desde cero una herramienta de diagramación para el modelado de la replicación de MySQL es una tarea compleja que consumiría mucho tiempo, es por ello que se revisó en la literatura las técnicas existentes para simplificar y acelerar el desarrollo de este tipo de herramientas.

Es así como se identificó una gran publicación de herramientas de diagramación y editores gráficos, en la que su desarrollo fue basado en los paradigmas de la Ingeniería Dirigida por Modelos (MDE) debido a la madurez alcanzada por los frameworks y herramientas para MDE tales como Graphical Modeling Framework (GMF), Generic Eclipse Modeling System (GEMS), VSVMSDK antes denominado Microsoft DSL Tools, Generic Modeling Environment (GME), Epsilon, Eugenia [Kolovos+ 2010], entre otros.

MDE se centra en la explotación de modelos como la piedra angular del proceso de desarrollo de software [Gascueña+ 2012]. MDE propone especificar sistemas software mediante modelos a diferentes niveles de abstracción, de forma que los modelos de más alto nivel se transformen en otros de menor nivel hasta alcanzar tanto como sea posible un modelo ejecutable mediante la generación automática de código [Cetinkaya+ 2011] y [Jiménez 2012].

MDE está basado en un número de principios que involucra los conceptos de modelo, metamodelo, meta-metamodelo y transformaciones de modelos para la generación automática del código ejecutable durante el proceso de desarrollo de software [Cetinkaya+ 2011, Gascueña+ 2012]. En este sentido, los modelos y las transformaciones entre dichos modelos juegan un papel central en los procesos de desarrollo, convirtiéndose en la base de este paradigma [Jiménez 2012].

Las herramientas son desarrolladas para un dominio en particular, conocidos también como lenguajes de dominio específico (DSL). DSL no es un concepto nuevo, pero ha llegado a ser popular en los últimos años debido a un nuevo enfoque de desarrollo orientado al dominio que la ingeniería dirigida por modelos (MDE) ha introducido. Los modelos diseñados en un DSL pueden ser validados y posteriormente usados como entrada en un proceso de transformación modelo a modelo y/o modelo a texto, la cual producen automáticamente otros modelos o artefactos basados en texto tales como código fuente o documentación [Spanou 2012].

1.2 Problema

A nivel de modelado, diversas herramientas de diagramación, tales como Microsoft Visio [Visio 2013a], Dia [Dia 2013], draw.io [Draw.io 2013], entre otras, son usadas para diseñar modelos de replicación MySQL. Sin embargo, este tipo de herramientas no permiten validar automáticamente si un modelo de replicación de MySQL está libre de errores, lo que puede resultar tener documentación errónea de los modelos de replicación. Dichas herramientas tampoco permiten generar automáticamente a partir de un modelo de replicación MySQL los comandos mysqlreplicate de configuración. La falta de la funcionalidad antes mencionada conlleva a realizarlas de forma manual, la cual es una tarea tediosa, propensa a errores y que consume tiempo, más aún si el número de servidores MySQL del modelo es alto, con 15, 20, 25 o más servidores.

1.3 Objetivos

1.3.1 Objetivo General

Desarrollar una herramienta gráfica de diagramación para el Modelado de la Replicación de MySQL basada en la Ingeniería Dirigida por Modelos (MDE) de forma de permitir: (i) Validar automáticamente si un Modelo de Replicación MySQL es correcto, mostrando los errores en caso existan, y reducir en más del 87% el tiempo empleado al usar Microsoft Visio 2013 en la identificación y corrección de errores de un modelo de replicación con 25 servidores, y (ii) Generar automáticamente a partir de un modelo válido, los comandos mysqlreplicate de configuración, y reducir en más del 99% el tiempo empleado usando el método manual con Microsoft Visio 2013.

1.3.2 Objetivos Específicos

i. Estudiar la documentación oficial para el Modelado de la Replicación MySQL y revisar las herramientas existentes.
ii. Revisar artículos científicos sobre el desarrollo de herramientas basadas en la Ingeniería Dirigida por Modelos.
iii. Seleccionar y estudiar los frameworks, herramientas y lenguajes a ser utilizados en el desarrollo de la herramienta propuesta.
iv. Definir el metamodelo para el Modelado de la Replicación de MySQL.
v. Realizar experimentos para evaluar la herramienta desarrollada.

1.4 Justificación

1.4.1 Justificación Práctica

La replicación de base de datos entre servidores MySQL es la estrategia más común y económica para mejorar el rendimiento, escalabilidad y disponibilidad en las aplicaciones basadas en MySQL [MySQL 2013e] y [MySQL 2013g].

Las herramientas de modelado tienen una larga historia en la ayuda a los administradores de base de datos (DBA's) e ingenieros de software, a visualizar, documentar, comunicar, colaborar y crear sistemas con mayor claridad, precisión y confiabilidad [Sybase 2006], es por ello que el Modelado de la Replicación de MySQL es una tarea importante realizada por los DBA's.

Los DBA's tienen la necesidad de contar con una herramienta que les permita además de diagramar un Modelo de Replicación MySQL, validarlo para una rápida identificación y corrección de errores, y adicionalmente, generar automáticamente a partir de un modelo válido los comandos mysqlreplicate de configuración. Todo esto con el fin de minimizar tareas manuales, reducir tiempo y esfuerzo que puede ser empleado en otras actividades críticas propias de un DBA en una organización.

1.4.2 Justificación Teórica

A nivel de modelado, diversas herramientas de diagramación, tales como Microsoft Visio [Visio 2013a], Dia [Dia 2013], draw.io [Draw.io 2013], entre otras, son usadas para diseñar modelos de replicación MySQL. Sin embargo, este tipo de herramientas no permiten validar automáticamente si un modelo de replicación MySQL es correcto, identificando y mostrando los errores en caso existan. Tampoco permiten generar automáticamente a partir de un modelo de replicación válido, los comandos mysqlreplicate de configuración.

La falta de la funcionalidad antes mencionada conlleva a realizarla de forma manual, la cual se torna en una tarea tediosa, propensa a errores y que consume tiempo, más aún si el número de servidores MySQL del modelo el alto, con 15, 20, 25 o más servidores.

La herramienta propuesta será desarrollada basada en la Ingeniería Dirigida por Modelos (MDE).

1.5 Organización de la Tesis

El presente trabajo de tesis de investigación está organizado en seis capítulos.

En el Capítulo 1 se realiza la introducción del trabajo de investigación, indicando claramente el problema, objetivo y justificación de la investigación.

En el Capítulo 2 se desarrolla el marco teórico conceptual en el que se describen los conceptos de la replicación de MySQL y los conceptos relacionados con la Ingeniería Dirigida por Modelos (MDE).

En el Capítulo 3 se revisa el estado del arte de las herramientas existentes para el Modelado de la Replicación de MySQL, así como del desarrollo de herramientas basadas en la Ingeniería Dirigida por Modelos (MDE).

En el Capítulo 4 se detalla el proceso de desarrollo de la herramienta propuesta.

En el Capítulo 5 se describen experimentos realizados para evaluar la herramienta desarrollada y confirmar si se cumple con el objetivo de la tesis. Finalmente se muestran los resultados obtenidos.

En el Capítulo 6 se presentan las conclusiones de la investigación y los trabajos futuros.

Capítulo 2: Marco Teórico

En este capítulo se va a definir los conceptos relevantes para esta investigación. El capítulo se inicia describiendo los conceptos de la replicación de MySQL. Luego se describe la teoría relacionada con la Ingeniería Dirigida por Modelos (MDE) que es la técnica seleccionada para el desarrollo de la herramienta propuesta en este trabajo de tesis de maestría. En el siguiente capítulo se presentará la revisión del estado del arte relacionado con esta investigación.

2.1 Replicación en MySQL

La replicación de base de datos es el proceso de mantener múltiples copias de un servidor principal en diferentes ubicaciones físicas llamadas réplicas [Plattner 2006], esta técnica es ampliamente usada principalmente para mejorar el rendimiento, escalabilidad y disponibilidad de los servidores de base de datos [Wiesmann+ 2000], [Al-Ekram+ 2010] y [Araújo 2011].

La replicación de base de datos mejora el rendimiento al reducir los tiempos de respuesta, esto se logra realizando un balanceo de carga de las transacciones entre todas las réplicas. También incrementa la disponibilidad mejorando la tolerancia a fallos, si el servidor principal falla debido a un error o por propósitos de mantenimiento, la información puede seguir siendo accesible en otras réplicas [Elnikety+ 2005], [Plattner 2006], [Cecchet+ 2008] y [Araújo 2011].

MySQL soporta la replicación desde su versión 3.23.15 [MySQL 2013c], que se liberó el 8 de Mayo de 2000 [MySQL 2013d]. En la replicación de MySQL, a un servidor principal se le conoce como maestro, y a una réplica se le denomina esclavo [MySQL 2013b], por lo consiguiente la replicación en MySQL permite duplicar los datos desde un servidor maestro hacia uno o más servidores esclavos.

MySQL soporta nativamente la replicación como una característica estándar del motor de base de datos, dependiendo de la configuración, se puede replicar todas las bases de datos, algunas bases de datos, o incluso algunas tablas de una base de datos [Araújo 2011].

El servidor maestro registrará en su log binario todos los cambios en los datos y objetos de la base de datos, luego del registro en el log binario los datos son enviados y aplicados en los servidores esclavos. En la Figura 2-1 se puede observar el flujo de trabajo interno de la replicación de MySQL entre un servidor maestro y un esclavo [MySQL 2013e].

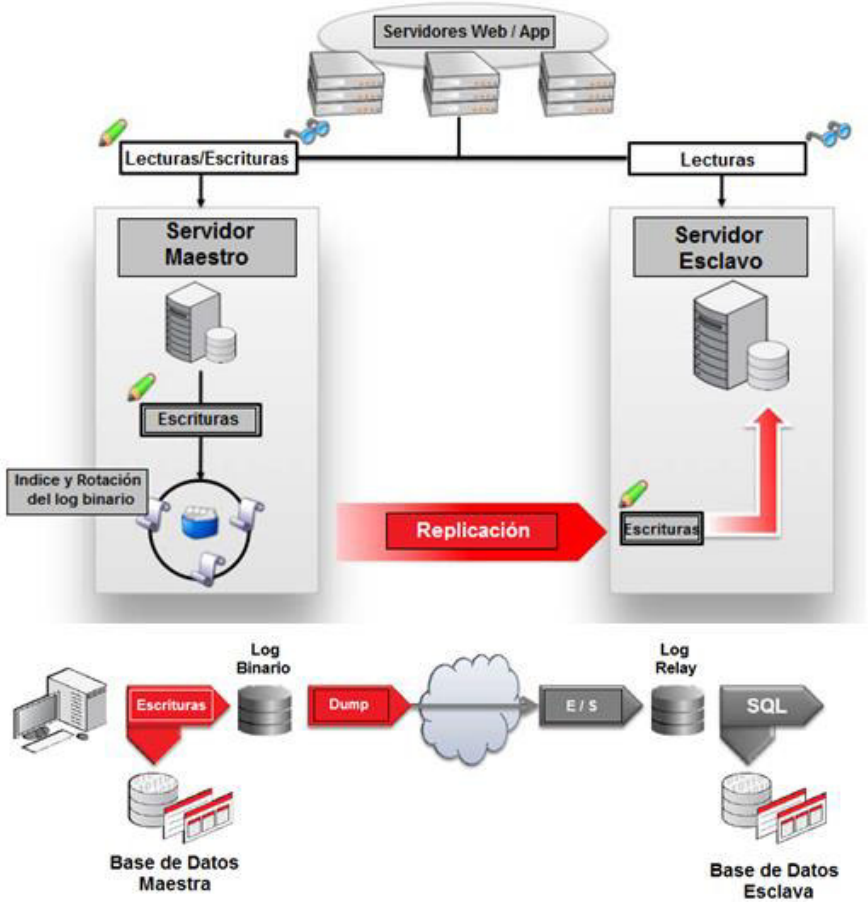


Figura 2-1 Flujo de Trabajo Interno de la Replicación de MySQL [MySQL 2013e]

Desde la versión 5.6.5 de MySQL, la replicación es basada en identificadores globales de transacción (GTIDs), la cual mejoró la consistencia entre un servidor maestro y sus esclavos [MySQL 2013b]. Un GTID es un identificador único que comprende el identificador UUID del servidor y un número de transacción, estos son automáticamente generados como la cabecera de cada transacción y escrito junto con la transacción en el log binario.

El uso de GTIDs hace simple el seguimiento y comparación de transacciones replicadas entre un servidor maestro y sus servidores esclavos, la cual habilita realizar un simple mecanismo de failover ante fallas del maestro. El motor de almacenamiento InnoDB debe ser usado con GTIDs para obtener todos los beneficios de alta disponibilidad.

2.2 Modelado de la Replicación de MySQL

Es posible configurar la replicación para casi cualquier topología de servidores maestros y esclavos, con la limitación de que un servidor esclavo puede tener un solo servidor maestro. En las siguientes secciones describiremos cada una de las topologías.

2.2.1 Maestro a Esclavo

Esta topología es la más popular y fácil de configurar y administrar, en esta topología se tiene dos servidores, uno que actúa como servidor maestro y otro como servidor esclavo. Todas las transacciones de escritura son realizadas en el servidor maestro y las operaciones de lectura pueden ser realizadas tanto en el servidor maestro como en el servidor esclavo. El modelo de esta topología es mostrado en la Figura 2-2.

Figura 2-2 Topología de Replicación Maestro a Esclavo [MySQL 2013e]

2.2.2 Maestro a Esclavos

En esta topología un servidor maestro tiene configurado varios servidores esclavos. Esto permite un mayor grado de escalabilidad pero tiene un costo mayor en administración. Los servidores esclavos presentan muy poca sobrecarga al servidor maestro, típicamente 1% de sobrecarga por cada servidor esclavo. La Figura 2-3 muestra el modelo de esta topología.

Figura 2-3 Topología de Replicación Maestro a Esclavos [MySQL 2013e]

2.2.3 Replicación Jerárquica o en Cascada

Esta configuración es una extensión de las topologías anteriores. En este caso, un servidor esclavo es asignado a otro servidor esclavo que actúa tanto como servidor maestro y esclavo puesto que está asignado a un servidor maestro principal. En esta configuración, todas las escrituras son realizadas en el servidor maestro principal. El modelo de esta topología es mostrada en la Figura 2-4.

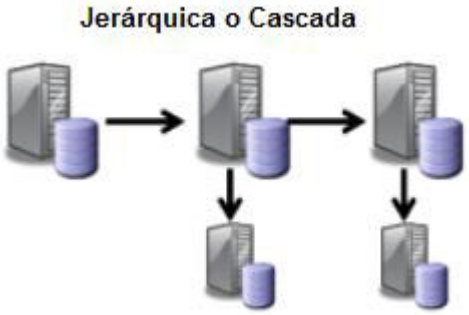


Figura 2-4 Topología de Replicación Jerárquica o en Cascada [MySQL 2013e]

2.2.4 Maestro a Maestro

En esta configuración, ambos servidores actúan como servidor maestro y esclavo. Se tiene la ventaja de poder realizar escrituras en cualquier servidor, pero esto aumenta el grado de complejidad en la instalación, configuración y administración. En esta topología no existe una detección y resolución de conflictos por lo que la aplicación debe asegurarse de no actualizar una fila mientras exista un cambio en la misma fila del otro servidor que no haya sido replicado aún.

Al usar esta topología, las columnas autoincrement deben ser usadas con los parámetros de configuración auto_increment_offset y auto_increment_increment para asegurarse de que no existan valores duplicados. La Figura 2-5 muestra el modelo de esta topología.

Figura 2-5 Topología de Replicación Maestro a Maestro [MySQL 2013e]

2.2.5 Replicación Circular

En esta topología cada servidor actúa como servidor maestro y esclavo. En este caso también es necesario utilizar los parámetros de configuración auto_increment_offset y auto_increment_increment para evitar valores duplicados en las columnas autoincrement. La Figura 2-6 muestra el modelo de esta topología.

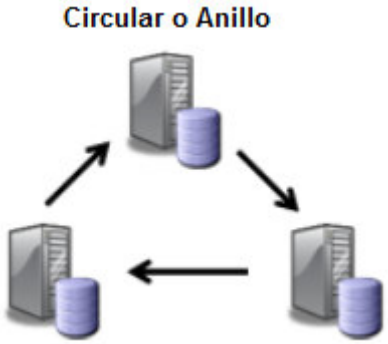


Figura 2-6 Replicación Circular [MySQL 2013e]

2.3 Configuración de la Replicación de MySQL

La configuración de la replicación de MySQL entre un servidor maestro y un servidor esclavo es una tarea sencilla, para lograr ello, un administrador de base de datos (DBA) o desarrollador en MySQL debe realizar lo siguiente:

i. Establecer los parámetros relacionados con la replicación en el archivo de configuración tanto del servidor maestro como del servidor esclavo de MySQL. La documentación oficial de MySQL indica que como mínimo se tiene que establecer parámetros para las siguientes opciones de la sección [mysqld] del archivo de configuración de cada servidor MySQL participante de la replicación:

- **binlog-format**: establecida en ROW (replicación basada en filas) para tomar ventaja de todas las optimizaciones de MySQL 5.6.
- **log-slave-updates**: establecida en ON para habilitar los identificadores globales de transacción (GTID's) y reunir los requisitos asociados.
- **gtid-mode**: establecida en ON para habilitar los identificadores globales de transacción (GTID's) y reunir los requisitos asociados.
- **enforce-gtid-consistency**: establecida en ON para habilitar los identificadores globales de transacción (GTID's) y reunir los requisitos asociados.

- **master-info-repository**: establecida en TABLE para almacenar la información relacionada al maestro en tablas transaccionales.
- **relay-log-info-repository**: establecida en TABLE para almacenar la información relacionada al esclavo en tablas transaccionales.
- **sync-master-info**: establecida en 1 para asegurar que la información no sea perdida.
- **slave-parallel-workers**: para establecer el número de hilos paralelos para ejecutar eventos de replicación.
- **binlog-checksum**: usada para habilitar la comprobación de la replicación.
- **master-verify-checksum**: usada para habilitar la comprobación de la replicación.
- **slave-sql-verify-checksum**: usada para habilitar la comprobación de la replicación.
- **binlog-rows-query-log_events**: establecida en ON para registrar en el log binario la consulta SQL original cuando se usa la replicación basada en filas y simplificar la solución de los problemas.
- **log-bin**: el log binario tiene que estar habilitado para que un servidor pueda actuar como maestro.
- **server-id**: esta variable debe ser única entre todos los servidores y el rango de sus valores está entre 1 y 2^{32}.

Para una mayor consistencia se tiene que establecer en 1 las siguientes opciones:

- **innodb_flush_log_at_trx_commit**
- **sync_binlog**

Luego de modificar el archivo de configuración de cada servidor, se debe de reiniciar el servicio de MySQL para que los cambios surtan efecto.

Estas configuraciones deben ser hechas manualmente por el DBA y es un proceso complejo de automatizar.

ii. Escribir y ejecutar el comando **mysqlreplicate** en la herramienta MySQL Utilities. En el comando mysqlreplicate se indica como parámetros los datos del servidor maestro y esclavo, con esto es suficiente pero también se pueden indicar algunos otros parámetros.

Uno de los propósitos de esta investigación es generar automáticamente estos comandos de configuración a partir de un modelo de replicación y evitar así su escritura manual. La Figura 2-7 muestra el resultado de ejecutar este comando en la herramienta MySQL Utilities.

```
mysqlreplicate --master=root@black:3306 --slave=root@blue:3306
# master on black: ... connected.
# slave on blue: ... connected.
# Checking for binary logging on master...
# Setting up replication...
# ...done.
```

Figura 2-7 Comando de MySQL Utilities: mysqlreplicate [MySQL 2013i]

2.4 Modelo

Los modelos tienen una larga tradición en la ingeniería de software y han sido ampliamente usados en proyectos de desarrollo de software [Tekinerdogan+ 2013]. Los modelos ayudan a analizar y entender el sistema, permiten describir la arquitectura del software.

Los modelos facilitan la especificación, evolución y mantenimiento del sistema [Spanou 2012]. Los modelos permiten a los desarrolladores del software una comunicación más efectiva con los stakeholders [Dantra+ 2009], [Gascueña+ 2012] y [Jiménez 2012].

2.5 Metamodelo

En el contexto de la ingeniería dirigida por modelos, un modelo es una descripción de un sistema en un lenguaje bien-formado. Un lenguaje de modelado bien-formado es aquel que define los elementos que pueden formar parte de un modelo así como las relaciones que pueden darse entre estos elementos. Generalmente, esta información es recogida por el metamodelo del lenguaje [Jiménez 2012]. La idea básica de metamodelo es identificar conceptos generales en un problema de dominio dado y las relaciones usadas para describir los modelos [Gascueña+ 2012].

Un metamodelo es un modelo que permite establecer las condiciones para que los modelos expresados en un determinado lenguaje se consideren válidos. En algunos casos, un metamodelo es creado conforme al lenguaje que él mismo define y se denomina meta-metamodelo [Jiménez 2012]. Los modelos son especificados mediante instancias de metamodelos, y el metamodelo es una instancia del meta-metamodelo [Cetinkaya+ 2011]. La Figura 2-8 ilustra estos conceptos.

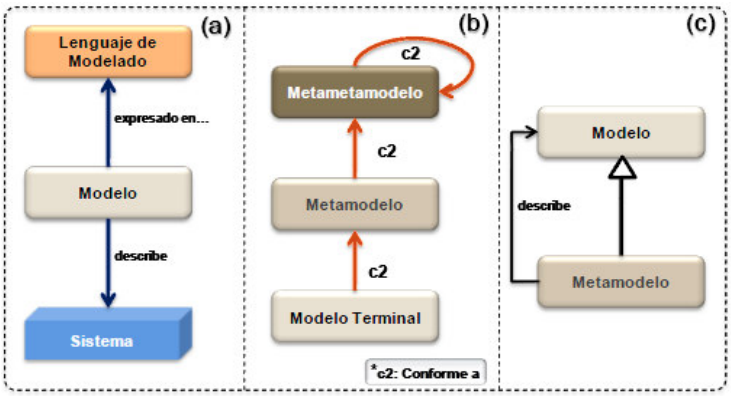


Figura 2-8 a) Definición de Modelo b) Jerarquía de Modelado y c) Relación entre los conceptos 'Modelo' y 'Metamodelo' [Jiménez 2012]

2.6 Transformación de Modelos

Una transformación de modelos es un proceso automatizado que toma uno o varios modelos origen y produce uno o varios modelos destino, a partir de la ejecución de un conjunto de reglas [Jiménez 2012].

Una transformación modelo a modelo (M2M) es usada para normalizar, optimizar y refactorizar modelos, y una transformación modelo a texto (M2T) es usada para generar código fuente o documentación desde los modelos.

2.7 Arquitectura Dirigida por Modelos

La Arquitectura Dirigida por Modelos (MDA) es un framework para el desarrollo de software presentado en el año 2001 por el Object Management Group (OMG) [Spanou 2012]. El principal objetivo de MDA es separar la especificación de un sistema concreto de los detalles relacionados con el uso de la plataforma por parte de dicho sistema. Para alcanzar este objetivo se centra en la definición de modelos formales como elementos de primera clase para el diseño e implementación de sistemas y la definición de transformaciones (semi-)automáticas entre los modelos [Jiménez 2012].

Uno de los rasgos distintivos de MDA es que define tres grandes grupos de modelos de acuerdo a su nivel de abstracción, los modelos independientes de computación (CIM, Computation Independent Model), los modelos independientes de la plataforma (PIM, Platform Independent Model) y los modelos específicos de la plataforma (PSM, Platform Specific Model).

Los requisitos del sistema se modelan mediante modelos independientes de computación (CIM, Computation Independent Model).

Los modelos independientes de la plataforma (PIM) sirven para representar la funcionalidad y estructura del sistema, omitiendo los detalles tecnológicos asociados a las características específicas de la plataforma que soportará al sistema.

Finalmente, las especificaciones descritas a nivel PIM son adaptadas a los detalles de plataformas concretas por medio de los modelos específicos de plataforma (PSM), a partir de los cuales se genera el código fuente del sistema [Jiménez 2012].

En el nivel PSM es posible definir modelos que representan distintos grados de abstracción, pudiéndose agrupar en modelos que describen elementos comunes y en modelos que contienen elementos específicos de una plataforma concreta [Cetinkaya+ 2011, Montenegro+ 2011, Jiménez 2012], estos modelos "intermedios", se denominan modelos dependientes de la plataforma (PDM, Platform Dependent Models) [Jiménez 2012].

La Figura 2-9 ilustra una simplificación de las relaciones entre las capas de abstracción definidas por MDA así como las interacciones existentes entre modelos de los diferentes niveles.

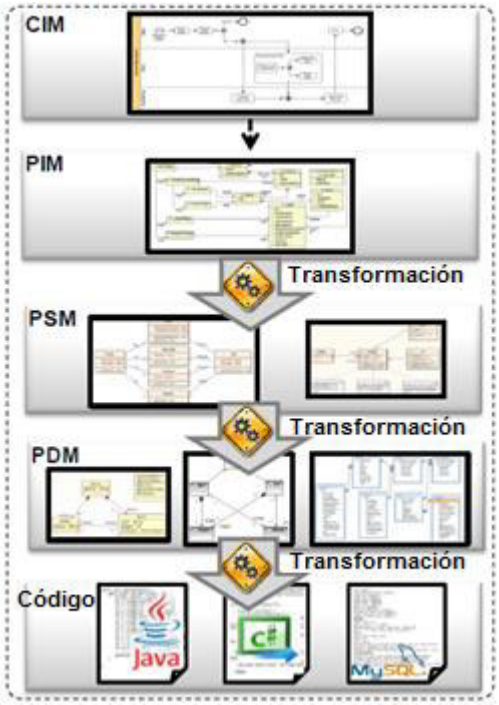


Figura 2-9 Niveles de Abstracción MDA [Jiménez 2012]

Otro rasgo distintivo de MDA es que potencia el uso de los estándares definidos por la OMG. Entre estos estándares se encuentran: Meta Object Facility (MOF), Object Constraint Language (OCL), XML Metadata Interchange (XMI) y Query/View/Transformation (QVT) [Cetinkaya+ 2011, Jiménez 2012].

2.8 Ingeniería Dirigida por Modelos

La Ingeniería Dirigida por Modelos (MDE) se centra en la explotación de modelos como la piedra angular del proceso de desarrollo de software [Gascueña+ 2012]. MDE propone especificar sistemas software mediante modelos a diferentes niveles de abstracción, de forma que los modelos de más alto nivel se transformen en otros de menor nivel hasta alcanzar tanto como sea posible un modelo ejecutable mediante la generación automática de código [Cetinkaya+ 2011] y [Jiménez 2012].

MDE está basado en número de principios que involucra los conceptos de modelo, metamodelo, meta-metamodelo y transformaciones de modelos para la generación automática del código ejecutable durante el proceso de desarrollo de software [Cetinkaya+ 2011, Gascueña+ 2012]. En este sentido, los modelos y las transformaciones entre dichos modelos juegan un papel central en los procesos de desarrollo, convirtiéndose en la base de este paradigma [Jiménez 2012].

MDE aprovecha los modelos para permitir a los desarrolladores una comunicación más efectiva con los stakeholders [Dantra+ 2009], [Gascueña+ 2012] y [Jiménez 2012]. MDE permite invertir más tiempo en el análisis de los requisitos del sistema (espacio del problema) y menos tiempo en la escritura de código (espacio de la solución) [Gascueña+ 2012] y [Jiménez 2012], reduciendo así tiempo y esfuerzo en el desarrollo de software [Dantra+ 2009], [Montenegro+ 2011], [Rodrigues+ 2011] y [Gascueña+ 2012]. De este modo, la calidad del código es mejorada [Dantra+ 2009].

MDE nació con el lanzamiento de MDA (Model Driven Architecture) en el año 2001 por el consorcio OMG (Object Management Group) [Ameller 2009, Jiménez 2012] y con la progresiva unificación de iniciativas que usan modelos como artefacto principal en el desarrollo de software, todas las iniciativas dirigidas por modelos tienen una gran cantidad de terminología con un particular significado [Ameller 2009]. En muchas de las siglas de estas iniciativas aparecen las letras MD para referirse a la frase en inglés model driven [Voelter, 2009, Quintero+ 2011].

MDE ha evolucionado y propone amplias definiciones no limitadas al desarrollo de software [Ameller 2009]. MDE es implementado por diferentes estándares e iniciativas como MDA, MIC (Model Integrated Computing), MS/DSLs (Microsoft Domain Specific Languages) y muchos otros.

MDE, MDD (Model Driven Development) y MDSD (Model Driven Software Development) son aproximaciones de desarrollo similares, mientras que MDA es una de las modalidades para aplicar e implementar MDE que usa los estándares OMG [Quintero+ 2011, Whittle+ 2013], la cual establece una serie de tecnologías a utilizar en la construcción de software bajo el esquema de MDE [Montenegro+ 2011]. La Figura 2-10 muestra las diferentes formas e iniciativas MDE.

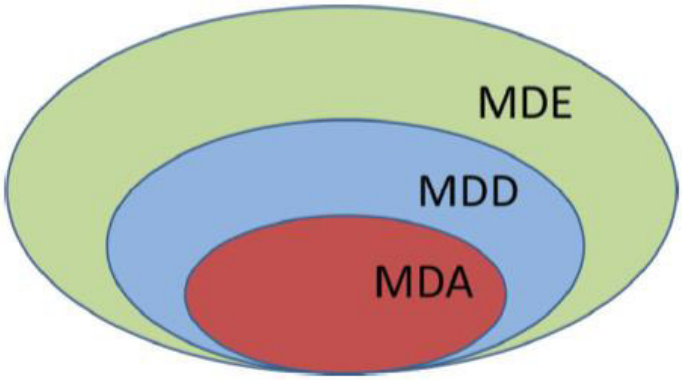


Figura 2-10 Diferentes Formas e Iniciativas de MDE [Whittle+ 2013]

Un enfoque MDE requiere una definición concisa y específica de los modelos, así Lenguajes de Dominio Específico (DSL) son frecuentemente usados.

Un DSL es un lenguaje dedicado a un dominio en particular y es usado para describir el lenguaje en el que se expresan los modelos.

Los modelos basados en un DSL son luego usados como entrada para una transformación modelo a modelo y/o modelo a texto, la cual producen otros modelos o artefactos basados en texto tales como código o documentación.

Se requieren de editores gráficos para poder visualizar los modelos expresados en un DSL [Spanou 2012]. Crear un modelo en una herramienta gráfica es rápido de crear, editar y mantener, comparado a la escritura de código [Sybase 2006]. Actualmente existen varios frameworks disponibles para desarrollar editores gráficos para DSLs [Spanou 2012].

2.9 Lenguaje de Dominio Específico

Un lenguaje de dominio específico (DSL), es un lenguaje especializado de alto nivel la cual se enfoca en un problema de dominio en particular.

Modelos basados en DSLs pueden ser usados como datos de entrada para transformaciones modelo a modelo o modelo a texto para la generación de código fuente o documentación.

DSL no es un concepto nuevo, pero ha llegado a ser popular en los últimos años debido a un nuevo enfoque de desarrollo orientado al dominio que la ingeniería dirigida por modelos (MDE) ha introducido.

DSLs son simples y fáciles de entender puesto que usa una sintaxis personalizada para su representación. Un buen DSL puede incrementar la productividad simplificando el código y mejora la comunicación con los expertos del dominio.

Los principales conceptos que son usados para construir un DSL son la sintaxis abstracta, la cual es usada para capturar la información de cada dominio específico, y la sintaxis concreta que especifica como el modelo capturado puede ser visualizado mejorando la comprensión del problema del dominio y facilitando la interacción con el modelo. Es importante, antes de iniciar el desarrollo de un DSL, adquirir un conocimiento maduro del dominio.

El modelado de dominio específico (DSM) es una metodología de ingeniería de software que aprovecha los lenguajes de dominio específico (DSL) para construir un sistema software. En tales enfoques, en lugar de usar técnicas de modelado de propósito general como UML en el que los conceptos de los modelos están en varios dominios, el modelado gira en torno a un dominio en específico.

El objetivo es mejorar la productividad de los desarrolladores y la calidad de los productos manteniendo un alto nivel de abstracción y automatizando la producción de software.

Los frameworks DSM han sido adaptados en IDEs, tales como Eclipse con Eclipse Modeling Framework (EMF) y Graphical Modeling Framework (GMF), y Microsoft DSL Tools ahora denominado Microsoft Visual Studio Visualization and Modeling SDK (VSVMSDK) [Spanou 2012].

2.10 Resumen del Capítulo

En este capítulo se ha descrito el marco teórico conceptual relevante para este trabajo de tesis de investigación. Se ha iniciado con la descripción de los conceptos de la replicación de MySQL. Se ha definido también las topologías de replicación soportadas por MySQL, las cuales deben poder ser modeladas con la herramienta propuesta y validar sus restricciones.

Luego se describieron los conceptos relacionados con la teoría de la Ingeniería Dirigida por Modelos (MDE), que es la técnica con la que se va a construir la herramienta propuesta en este trabajo de tesis de maestría. Específicamente se definió a un modelo, metamodelo, las transformaciones entre modelos, la arquitectura dirigida por modelos (MDA), lenguaje de dominio específico (DSL) y el modelado de dominio específico (DSM).

En el siguiente capítulo se presentará la revisión del estado del arte relacionado con esta investigación.

Capítulo 3: Estado del Arte de Herramientas para el Modelado de la Replicación de MySQL y de Herramientas Desarrolladas Basadas en la Ingeniería Dirigida por Modelos (MDE)

En este capítulo se presentan las principales herramientas existentes que contribuyen al estado del arte de esta tesis de investigación. El capítulo se inicia con un resumen de la revisión de la literatura realizada, luego se realiza la descripción de las herramientas existentes para el modelado de la replicación de MySQL. Finalmente se describe el desarrollo de ocho herramientas seleccionadas basadas en la Ingeniería Dirigida por Modelos (MDE). El detalle del desarrollo de la herramienta propuesta en esta tesis de maestría se describe en el siguiente capítulo.

3.1 Revisión de la Literatura

Los administradores de base de datos (DBA's) y profesionales en MySQL utilizan herramientas de diagramación de propósito general tales como Microsoft Visio [Visio 2013a], Dia [Dia 2013], Draw.io [Draw.io 2013], Gliffy [Gliffy 2014], ConceptDraw [ConceptDraw 2014], Creately [Creately, 2014], FreelyDraw [FreelyDraw, 2014], SmartDraw [SmartDraw, 2014], LucidChart [LucidChart, 2014], etc, para diagramar sus modelos de replicación de MySQL. Sin embargo, en la revisión de la literatura no se encontró ninguna herramienta liberada como software libre o comercial, ni publicada como propuesta académica, que permita validar automáticamente un modelo de replicación de MySQL y que permita generar automáticamente los comandos de configuración a partir de un modelo de replicación de MySQL.

Existen varias herramientas diseñadas exclusivamente para configurar, administrar y monitorear la replicación de MySQL, entre las principales herramientas de configuración y administración tenemos a MySQL Utilities que es la herramienta oficial de MySQL, Openark Kit, MHA y Percona Toolkit, todas éstas herramientas son de línea de comandos (CLI), y entre las principales herramientas de monitoreo tenemos a MySQL Enterprise Monitor y MySQL Performance Monitor [Bradford+ 2012].

Desarrollar a la medida y desde cero una herramienta de diagramación para el modelado de la replicación entre servidores MySQL es una tarea compleja que consumiría mucho tiempo, es por ello que se revisó en la literatura sobre técnicas y métodos para su desarrollo, así identificamos que la Ingeniería Dirigida por Modelos (MDE) propone especificar sistemas software mediante modelos a diferentes niveles de abstracción, de forma de que los modelos de más alto nivel se transformen en otros de menor nivel hasta alcanzar tanto como sea posible un modelo ejecutable mediante la generación automática de código [Cetinkaya+ 2011] y [Jiménez 2012].

La madurez de los frameworks y herramientas basados en MDE facilitan el desarrollo de editores gráficos para visualizar modelos expresados en un lenguaje de dominio específico (DSL). Un DSL es un lenguaje dedicado a un dominio en particular. Los modelos diseñados en un editor gráfico DSL pueden ser usados como entrada en un proceso de transformación modelo a modelo y/o modelo a texto, la cual producen automáticamente otros modelos o artefactos basados en texto tales como código fuente o documentación.

Un entorno de metamodelamiento es un medio usado para desarrollar una herramienta de modelamiento mediante la definición de su metamodelo en un meta-metamodelo predefinido. Los lenguajes de metamodelado más populares son la especificación Meta Object Facility (MOF), UML, Eclipse Modeling Framework (EMF) (lenguaje Ecore) y MetaGME [Cetinkaya+ 2011]. EMF, es la plataforma de metamodelado más adoptada en el ámbito de MDE [Jiménez+ 2010]. Ecore es el metamodelo usado por EMF para definir metamodelos. Ecore es una poderosa herramienta para diseñar arquitecturas dirigidas por modelos (MDA). Ecore es una implementación del lenguaje EMOF (Essential MOF), que es un subconjunto de MOF. El modelo EMOF provee de un conjunto mínimos de elementos requeridos para especificar metamodelos. Los principales elementos de Ecore son EClass, EReference y EAttribute.

Existen diferentes frameworks y herramientas de soporte a la ingeniería dirigida por modelos (MDE) para crear editores gráficos para DSLs, como por ejemplo Microsoft Domain Specific Language (DSL) Tools, MetaEdit+, [Dantra+ 2009] y [Cetinkaya+ 2011], Marama [Dantra+ 2009], Eclipse Graphical Modeling Framework (GMF) [Dantra+ 2009], [Jiménez+ 2010] y [Cetinkaya+ 2011], GenGMF, KybeleGMFgen [Jiménez+ 2010], Generic Modeling Environment (GME) [Cetinkaya+ 2011] y [Lau+ 2012], Generic Eclipse Modeling System (GEMS) [Cetinkaya+ 2011], Eugenia

[Kolovos+ 2010] y [Tekinerdogan+ 2013]. GMF es uno de los frameworks más usados para generar editores gráficos para crear modelos de acuerdo al metamodelo especificado. De esta forma, el código fuente es automáticamente generado usando como datos de entrada el modelo especificado en el editor gráfico [Gascueña+ 2012].

Object Constraint Language (OCL) puede ser usado para definir restricciones sobre los metamodelos, las restricciones aplican a todos los modelos que son instancias del metamodelo. Epsilon Validation Language (EVL) [Kolovos+ 2009b] permite definir restricciones de forma similiar al lenguaje OCL.

Para la generación de código existen varias tecnologías, tales como Acceleo, Xpand, MOFScript [Rodrigues+ 2011], JET (Java Emitter Templates) [Gascueña+ 2012], Epsilon Generation Language (EGL) [Rose+ 2008], todas emplean el mismo principio, la creación de reglas de transformación basándose en un metamodelo. Estas reglas serán aplicadas al modelo, para generar el código fuente en el lenguaje deseado, este tipo de tecnologías reciben el nombre de transformaciones modelo a texto (M2T, Model to Text). La transformación de un modelo en otro modelo se le conoce como transformación modelo a modelo (M2M, Model to Model), entre las herramientas que realizan esta tarea tenemos a ATL (ATLAS Transformation Language), Operational QVT [Jiménez 2012] y [Montenegro+ 2011] y Epsilon Transformation Language (ETL) [Kolovos+ 2008].

A continuación se describe brevemente el desarrollo de algunas herramientas gráficas de modelado de diferentes dominios basadas en MDE.

[Dantra+ 2009] desarrollaron una herramienta gráfica para el modelado de reportes y generación de código. Para la construcción de la herramienta, los autores utilizaron Microsoft DSL Tools, ahora denominado Microsoft Visual Studio Visualization and Modeling SDK (VSVMSDK), seleccionaron esta herramienta de desarrollo debido a que el motor de reportes es una solución basada en Microsoft. Para especificar restricciones al metamodelo utilizaron DSL Tools y funciones C#, y para la generación de código fuente usaron el motor de plantillas T4.

[Jiménez+ 2010] propuso KybeleGMFgen (Kybele GMF Generator) para soportar la generación automática de diagramadores para modelos de Eclipse Modeling Framework (EMF). La solución está basada en Graphical Modeling Framework (GMF) y la herramienta Eugenia. Fue aplicado un caso de estudio para el modelado de Bases de

Datos Objeto-Relacionales en Oracle 10g. En particular se obtiene un diagramador para modelar tipos estructurados y tablas tipadas.

[Cetinkaya+ 2011] presentaron una herramienta de soporte para un entorno de modelado y simulación. La herramienta está basada en EMF. Generic Eclipse Modeling System (GEMS) fue utilizado para implementar el editor gráfico. Se desarrollaron transformaciones M2M escritas en ATL. Se definió una aplicación Java para generar el código fuente.

[Montenegro+ 2011] realizaron un editor gráfico para la construcción de módulos en sistemas de gestión del aprendizaje (LMS). El metamodelo se desarrolló con el lenguaje Ecore de EMF, se usó GMF para la construcción del editor gráfico de modelado. MOFScript fue usado para realizar las transformaciones modelo a texto de generación de código para el LMS seleccionado.

[Rodrigues+ 2011] implementaron una herramienta para el desarrollo de aplicaciones de redes inalámbricas de sensores. Desarrollaron sus metamodelos en Ecore. GenModel de EMF fue usado para crear el editor del modelo desde los metamodelos. Se usó Operational QVT para las transformaciones modelo a modelo (M2M). Se usó el plugin de eclipse Acceleo para las transformaciones modelo a texto (M2T).

[Gascueña+ 2012] desarrollaron una herramienta gráfica para la definición de modelos de una metodología para la construcción de sistemas multiagentes. El lenguaje Ecore de EMF es usado para especificar el metamodelo. GMF es usado para construir el editor gráfico para la creación de modelos gráficos. Java Emitter Templates (JET) es usado para crear plantillas para generar automáticamente el código.

[Lau+ 2012] construyeron una herramienta para soportar el desarrollo basado en componentes de acuerdo a un modelo de componentes que también propusieron. La herramienta fue implementada usando GME, tanto el metamodelo como el editor gráfico de modelado fueron realizados en GME. Se desarrollaron intérpretes en C++ para la generación de código.

[Tekinerdogan+ 2013] propusieron una herramienta de soporte para el modelo y análisis de puntos de vista de arquitectura de software. El metamodelo es definido en el lenguaje Ecore de EMF. Se utilizó Eugenia para la generación del editor gráfico desde un metamodelo Ecore con anotaciones. Para anotar el metamodelo Ecore con información de sintaxis concreta visual, se ha utilizado EMFatic.

3.2 Herramientas para el Modelado de la Replicación de MySQL

En esta sección se revisa el estado del arte de las herramientas existentes para el modelado de la replicación de MySQL.

Para diagramar un modelo de replicación de MySQL, los administradores de base de datos (DBA's) y profesionales en MySQL utilizan herramientas de diagramación de propósito general, tales como:

- Microsoft Visio [Visio 2013a]
- Dia [Dia 2013]
- Draw.io [Draw.io 2013]
- Gliffy [Gliffy 2014]
- ConceptDraw [ConceptDraw 2014]
- Creately [Creately, 2014]
- FreelyDraw [FreelyDraw, 2014]
- SmartDraw [SmartDraw, 2014]
- LucidChart [LucidChart, 2014], entre muchos otros más.

Sin embargo, en la revisión de la literatura no se encontró ninguna herramienta liberada como software libre o comercial, ni publicada como propuesta académica, que permita validar automáticamente un modelo de replicación de MySQL y que permita generar automáticamente los comandos de configuración a partir de un modelo de replicación de MySQL.

A continuación se va a describir brevemente las siguientes 3 herramientas de diagramación seleccionadas, Microsoft Visio 2013, Dia y Draw.io.

3.2.1 Microsoft Visio 2013

La herramienta Microsoft Visio 2013 es un software comercial de dibujo vectorial para Microsoft Windows [Visio 2013a].

Las herramientas que lo componen permiten realizar diagramas de oficinas, diagramas de bases de datos, diagramas de flujo de programas, UML, etc [Visio 2013a].

Microsoft Visio 2013 permite validar diagramas Flowchart y de BPMN 2.0, tal como se puede observar en la Figura 3-1.

Con esta herramienta podemos diagramar un modelo de replicación de MySQL, sin embargo, no permite validar automáticamente un modelo, ni generar los comandos de configuración a partir de un modelo.

La documentación de la herramienta Microsoft Visio indica que se puede extender su funcionalidad para programar validaciones complejas y personalizadas de diagramas [Visio 2013a], sin embargo, esta investigación pretende desarrollar una herramienta multiplataforma.

En la Figura 3-1 podemos observar la interfaz de la herramienta Microsoft Visio 2013 y un modelo de replicación de MySQL.

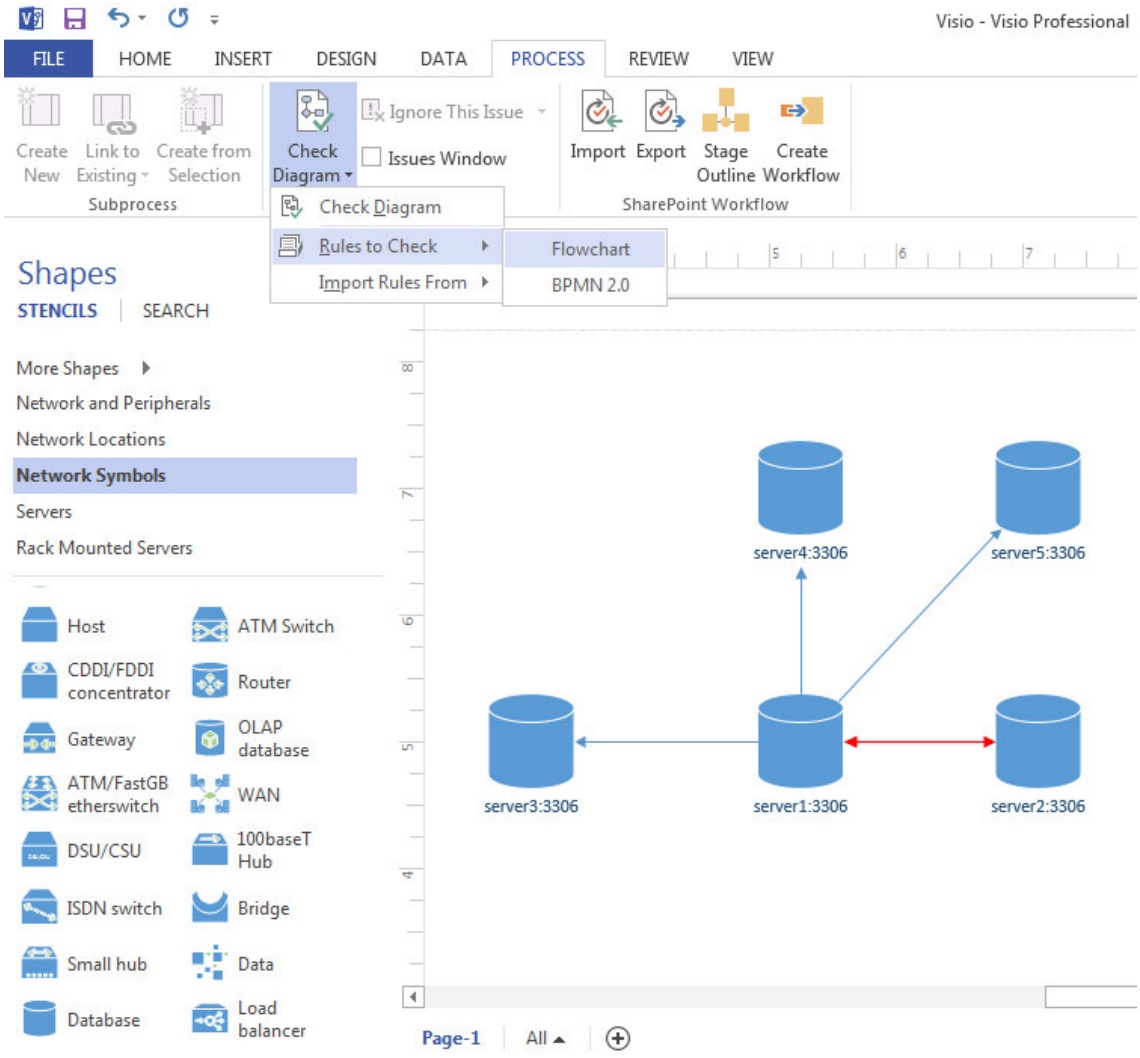


Figura 3-1 Microsoft Visio 2013 [Elaboración Propia]

3.2.2 Dia

Es una aplicación de software libre (open source) como alternativa a Microsoft Visio. Permite exportar los resultados a los formatos más populares de imágenes y diagramas, incluyendo los .vdx de Microsoft Visio. Funciona en Windows, GNU/Linux y Mac [Dia 2013]. En la Figura 3-2 se muestra la interfaz de esta herramienta.

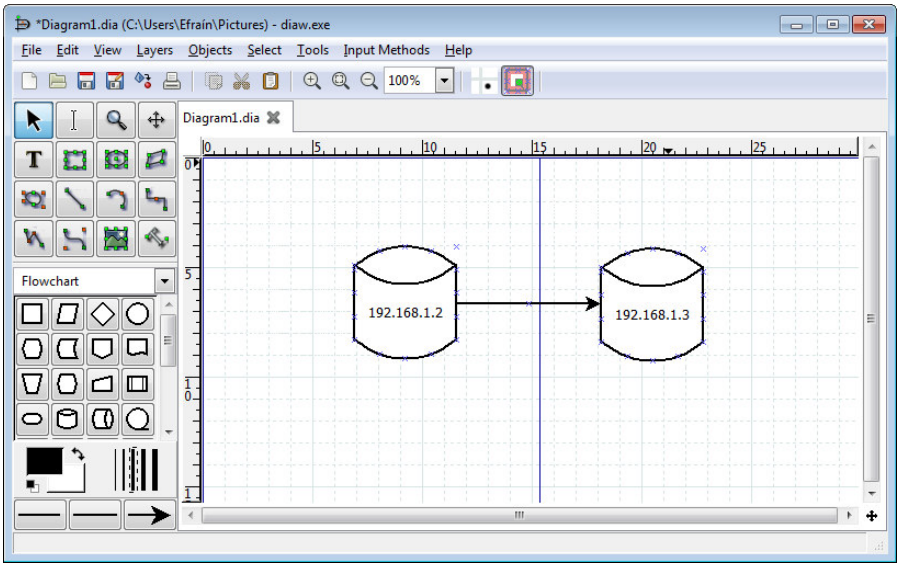


Figura 3-2 Herramienta Dia [Dia 2013]

3.2.3 Draw.io

Es una herramienta web como alternativa a la herramienta comercial Microsoft Visio, no requiere de registro para poder usarla, basta con entrar a la página para ya estar en el campo de dibujo [Draw.io 2013]. La Figura 3-3 muestra la interfaz de esta herramienta.

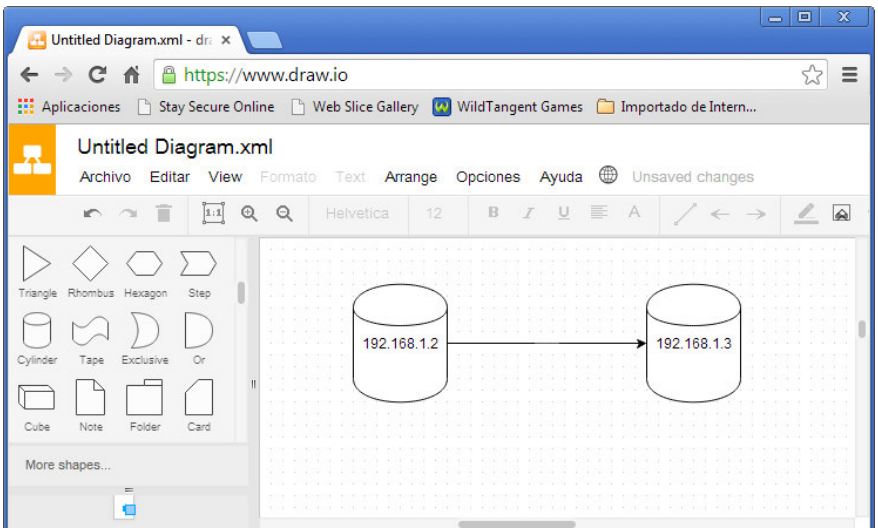


Figura 3-3 Herramienta Draw.io [Draw.io 2013]

3.3 Herramientas Desarrolladas Basadas en MDE

3.3.1 Diseño y Generación de Reportes

[Dantra+ 2009] presentan una herramienta gráfica para el diseño de reportes para el motor de reportes del ERP (Enterprise Resource Planning) Prism. Este motor de reportes permite a los usuarios finales escribir queries y presentar los resultados gráficamente como reportes.

Los autores proponen desarrollar modelar y diseñar los reportes en una herramienta gráfica para luego generar automáticamente desde el modelo los queries que deben ser ejecutados en el motor de reportes y permita a los usuarios implementar rápidamente un reporte. Esta herramienta reemplaza a un existente lenguaje textual proveyendo un enfoque más accesible y productivo de escribir reportes por usuarios finales principiantes e intermedios. El desarrollo de la herramienta se hizo en Microsft DSL (Domain Specific Language) Tools.

El aporte involucra 4 pasos. El primer paso es el diseño del metamodelo para el motor de reportes, parte del metamodelo propuesto por los autores aunque poco legible es mostrado en la Figura 3-4.

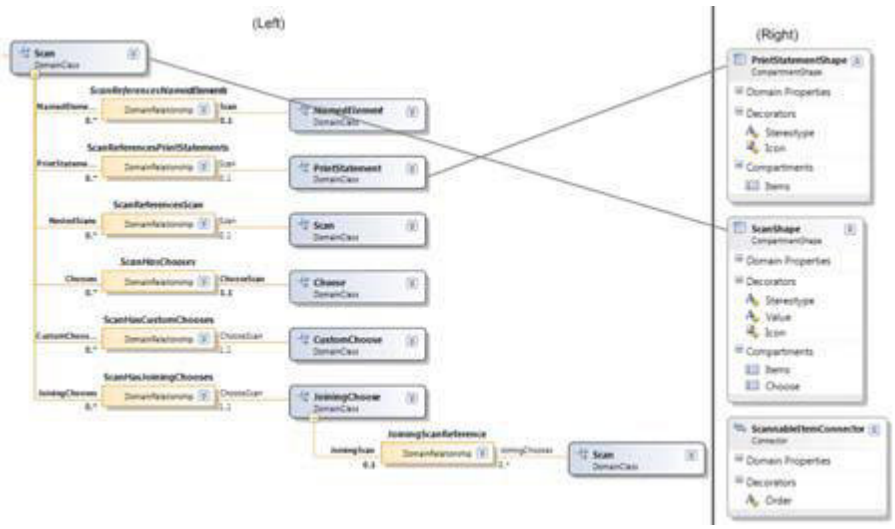


Figura 3-4 Metamodelo del Motor de Reportes [Dantra+ 2009]

El siguiente paso es la especificación de restricciones al metamodelo, esto provee las principales semánticas del lenguaje, se usó restricciones de DSL Tools y funciones C#. El tercer paso es la construcción de la herramienta gráfica basada en el metamodelo y

sus restricciones, especificando la forma visual de los elementos y relaciones del metamodelo.

Como último paso se usó el motor de plantillas T4 para desarrollar los generadores de código de los queries basados en el modelo, estos queries generados serán ejecutados manualmente en motor de reportes hacia la base de datos del ERP.

Los autores afirman que eligieron Microsoft DSL Tools ahora denominado Microsoft Visual Studio Visualization and Modeling SDK (VSVMSDK), debido a que el ERP Prism es una solución basada en Microsoft y porque provee una plataforma adecuada para entregar un robusto reporteador visual a usuarios finales. La Figura 3-5 muestra una parte de la interfaz de usuario de la herramienta.

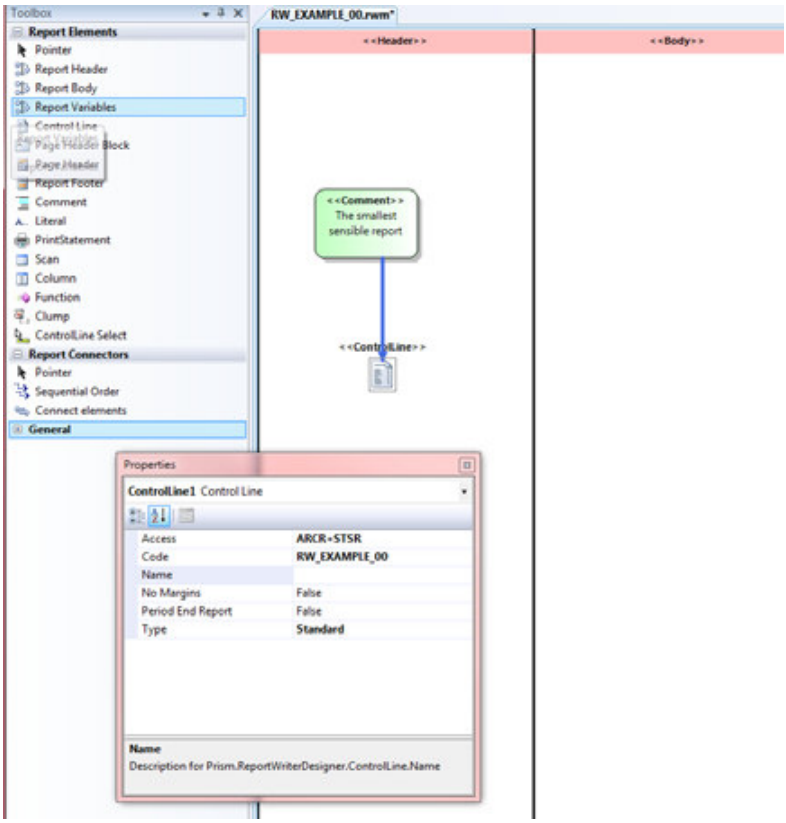


Figura 3-5 Interfaz de la Herramienta de Generación de Reportes [Dantra+ 2009]

La evaluación de la herramienta demuestra que su uso ayudó a los usuarios finales a evitar cometer errores.

Los usuarios determinaron también que el uso de la herramienta ayuda a ahorrar tiempo y esfuerzo y resulta ser efectiva para aumentar la productividad y precisión en la escritura de reportes, pues existe una eliminación casi completa de los errores ortográficos y de sintaxis reduciendo la propensión a errores.

3.3.2 Modelado de Bases de Datos Objeto-Relacionales en Oracle 10g

[Jiménez+ 2010] propone el desarrollo de una herramienta llamada KybeleGMFGen (Kybele GMF Generator) para soportar la generación automática de diagramadores para modelos EMF. Su solución, que está basada en GMF y Eugenia, pretende explotar las ventajas proporcionadas por estas herramientas y solucionar su principal desventaja, que no es otra que el número de tareas que el desarrollador debe realizar manualmente. Como caso de estudio se muestra el uso de KybeleGMFGen para la generación de un diagramador para una versión muy simplificada de un DSL gráfico para el modelado de Bases de Datos Objeto-Relacionales en Oracle 10g.

En particular se obtiene un diagramador para modelar tipos estructurados y tablas tipadas, de acuerdo al metamodelo simplificado mostrado en la parte izquierda de la Figura 3-6. Una vez que definen el metamodelo, añaden un conjunto de anotaciones GMF para dirigir el proceso de generación tal como muestra la parte derecha de la Figura 3-6, en estas anotaciones se indica que la representación de los objetos Model será el diagrama completo y que los objetos Structured Type y TypedTable se representarán como nodos en el diagrama, que incluirán unas etiquetas, etc.

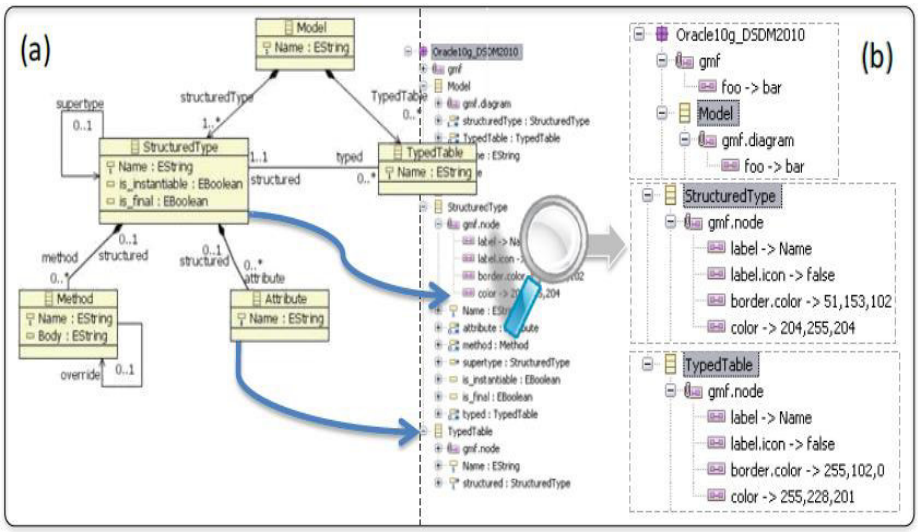


Figura 3-6 (a) Metamodelo de Modelado de Bases de Datos Objeto-Relacionales en Oracle 10g, (b) Vista Parcial del Metamodelo Anotado [Jiménez+ 2010]

Los autores indican que en este punto se puede generar directamente el diagramador, pero se obtendría un editor muy simple que solo permitiría "dibujar" cajas que representaran tipos estructurados y tablas tipadas. Para obtener un editor gráfico más complejo se debe de codificar los archivos de personalización (Ecore2GMF.eol,

FixGenModel.eol y FixGMFGen.eol). El archivo Ecore2GMF.eol se usa para definir algunos elementos gráficos, configurar la paleta de herramientas y especificar las correspondencias entre los elementos de los modelos GMF.

Para facilitar la tarea de codificar archivos de personalización los autores han elaborado una librería de operaciones comunes que se recogen en el archivo Operations.eol. Una vez codificado el archivo de personalización, se invoca la generación automática que soporta KybeleGMFgen, invocando la opción "Generate GMF editor code" desde el menú contextual del metamodelo ecore (Figura 3-7 (a)). La herramienta informará sobre el resultado del proceso y a partir de ese momento se puede utilizar el nuevo diagramador, tal como muestra la Figura 3-7 (b).

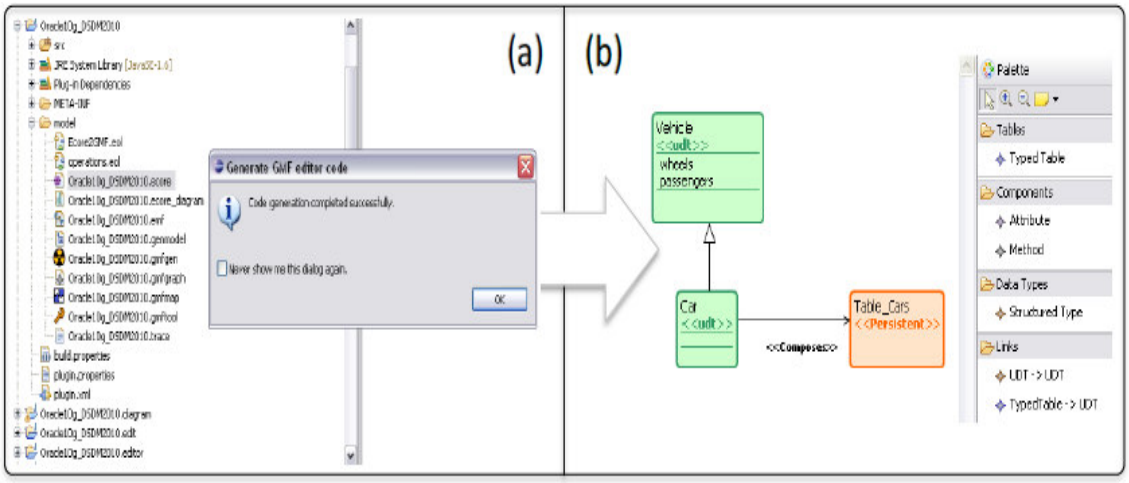


Figura 3-7 (a) Invocación de la Generación con KybeleGMFgen. (b) Ejemplo de Uso del Diagramador [Jiménez+ 2010]

Como todas las opciones de personalización están codificadas en los archivos EOL, el trabajo de personalización del editor no se perderá. No obstante, aunque la herramienta permite automatizar aproximadamente un 80% del proceso de desarrollo, no exime al desarrollador de conocer EMF y GMF si desea obtener diagramadores con funcionalidades muy concretas.

Los autores señalaron como trabajo futuro eliminar las anotaciones GMF del metamodelo, pues están contaminando el metamodelo con información que no es propia del dominio que representa. Se ha identificado que una posible desventaja de KybeleGMFgen es que necesita ser modificada y adaptada cada vez que se libere una nueva versión de GMF.

3.3.3 Modelado y Simulación de BPMN

[Cetinkaya+ 2011] indican que aunque la investigación en técnicas de diagramación y herramientas de soporte para modelado y simulación se ha incrementado en la última década, no hay muchos estudios que utilicen y obtengan provecho del enfoque dirigidos por modelos. Además muchas herramientas de simulación no cuenta con un soporte para el modelado conceptual. Existe una gran necesidad para automatizar y soportar el desarrollo de modelos de simulación a lo largo de todo su ciclo de vida, el cual reducirá el costo de desarrollo e incrementará la calidad.

Los autores proponen aplicar MDE en modelamiento y simulación para llenar el vacío entre las fases del modelado conceptual, especificación e implementación.

Los autores presentan un framework y una herramienta DSL (MDD4MS) para soportar todo el proceso. Para la implementación de la herramienta, los autores eligieron Eclipse, desde que esta plataforma está diseñada para la construcción de nuevos IDEs que pueden ser usados para crear aplicaciones. Además la comunidad eclipse provee de plugins para aplicar el enfoque dirigido por modelos y crear editores gráficos.

La herramienta está basada en EMF. Los metamodelos son definidos con el lenguaje de meta-metamodelo Ecore y los modelos son grabados como archivos XMI. BPMN (Business Process Model and Notation) fue seleccionado para la fase del modelado conceptual. El metamodelo simplificado de BPMN es mostrado en la Figura 3-8.

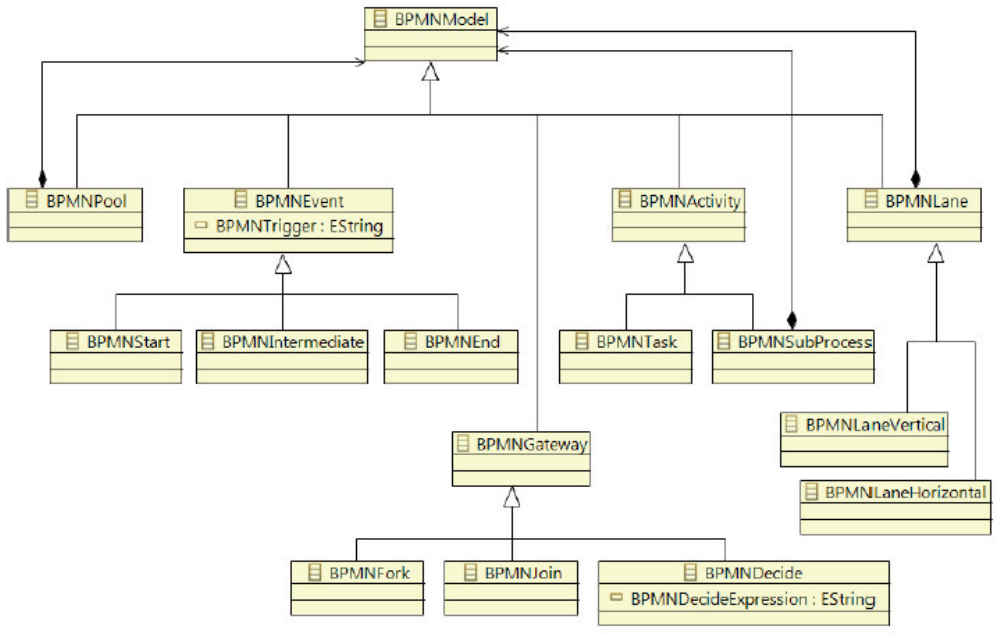


Figura 3-8 Metamodelo Simplificado de BPMN [Cetinkaya+ 2011]

El proyecto EMF es un framework de modelado para construir herramientas. De una especificación del modelo descrito en XMI, EMF provee herramientas y soporte de ejecución para producir un conjunto de clases Java para el modelo, junto con un conjunto de clases adaptadores que habilitan la vista y edición del modelo basado en comandos y un editor básico.

El core del framework EMF incluye un metamodelo (Ecore) para describir modelos y soporte su ejecución incluyendo notificaciones de cambios, soporte de persistencia con serialización XMI por defecto y un muy eficiente API para manipular objetos EMF genéricamente.

DEVS (Discrete Event System Specification), es un poderoso modelo de simulación que fue seleccionado para la fase de especificación. DSOL (Distributed Simulation Object Library) fue seleccionado para proveer la simulación y ejecución de funcionalidades. DEVDSOL está escrito en Java y usa DSOL, y fue seleccionado para la fase de implementación.

GEMS (Generic Eclipse Modeling System) es una parte del proyecto GMT (Generative Modeling Technologies). GEMS provee la habilidad de auto generar la implementación de una herramienta gráfica de modelado desde un metamodelo.

Transformaciones modelo a modelo (M2M) son escritas en ATL Transformation Language (ATL) ya que es uno de los lenguajes M2M más populares. Se definió una aplicación Java para generar el código para cada componente de DEVS. Los autores no mostraron resultados sobre la validación de la herramienta. La Figura 3-9 muestra la interfaz de usuario de la herramienta de modelado BPMN.

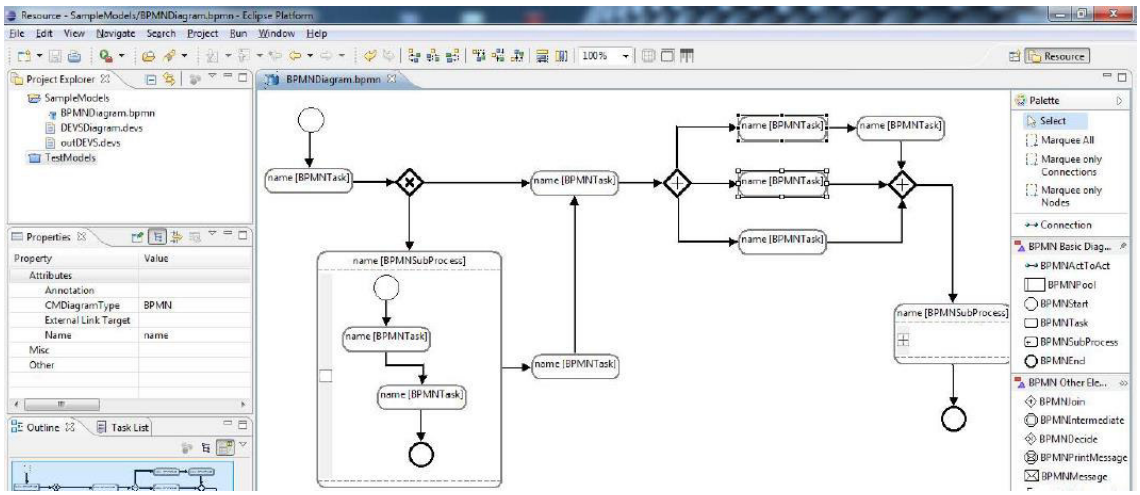


Figura 3-9 Interfaz de la Herramienta de Modelado BPMN [Cetinkaya+ 2011]

3.3.4 Modelado y Generación de Módulos LMS

[Montenegro+ 2011] presentan una herramienta para el modelado y generación de código para la construcción de módulos en los sistemas de gestión del aprendizaje (LMS) moodle, claroline y atutor.

El lenguaje de meta-metamodelo utilizado es el lenguaje Ecore que es el metamodelo que utiliza Eclipse. Para la utilización de ecore en Eclipse es necesario tener instalado el plugin de EMF (Eclipse Modeling Framework), este plugin provee básicamente dos herramientas para construir un modelo basado en ecore, una el Ecore Model que es un editor manual que funciona en un estilo de árbol de navegación para la creación del modelo basado en Ecore, la otra es el Ecore Diagram siendo este un editor gráfico similar a las herramientas gráficas para la creación de diagramas de clases UML.

Cualquiera de las dos formas que se utilice para crear el diagrama basado en Ecore, genera un fichero XMI que es una especificación para el intercambio de diagramas, en este caso se usó Ecore Model. El metamodelo LMS propuesto por los autores es mostrado en la Figura 3-10.

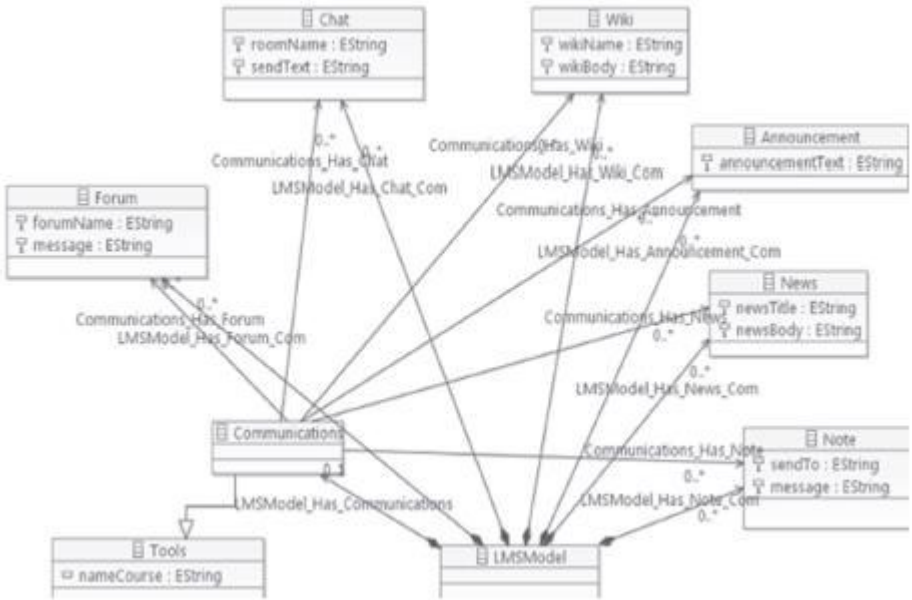


Figura 3-10 Metamodelo de LMS [Montenegro+ 2011]

Los autores emplearon Graphical Modeling Framework (GMF) para la construcción del editor gráfico, GMF es parte del proyecto GMP (Graphical Modeling Project) de Eclipse. Según el dashboard de GMF lo primero que se debe crear es el Domain Model, este corresponde al metamodelo LMS. El siguiente paso es crear el Domain Gen Model,

que es un modelo que permite transformar automáticamente el modelo Ecore a código Java. Luego se crea el Graphical Def Model, este es usado para definir las figuras, nodos, conexiones, etc.

El siguiente paso es la creación del Tooling Def Model, este es usado para especificar la paleta (Pallete) de herramientas de creación, acciones, etc, para los elementos gráficos. Allí existe un elemento en el nivel superior "Tool Registry", en el que se encuentra una paleta (Palette). La "Palette" contiene un "Tools Group" con elementos de tipo "Creation Tool" para los nodos y conexiones.

El "mapping model" es el siguiente paso, este combina los tres modelos: el "Domain Model", el "Graphical Def Model", y el "Tooling Def Model". El último paso es la creación del "Diagram Editor Gen Model", allí se establecen las propiedades para la generación de código. A partir de este modelo se obtiene un plugin para eclipse que contiene la herramienta construida. Se empleó MOFScript para la generación de código.

La apariencia de la herramienta se muestra en la Figura 3-11.

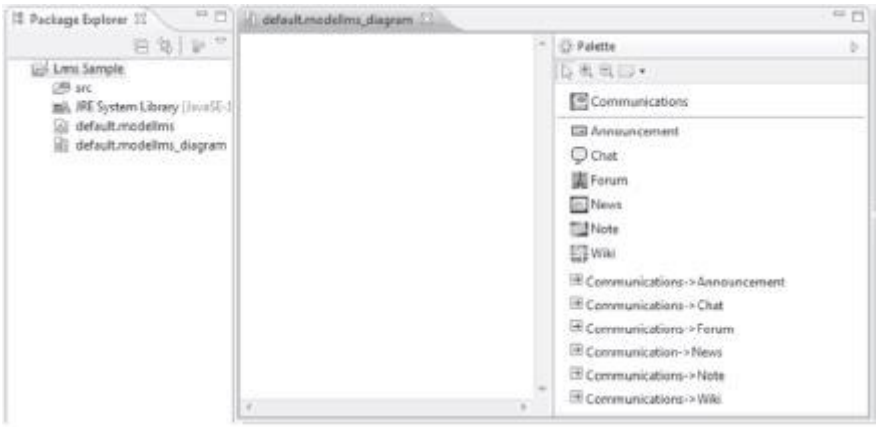


Figura 3-11 Interfaz de la Herramienta para Modelar Módulos en LMS [Montenegro+ 2011]

Las pruebas de evaluación de la herramienta midieron el tiempo y esfuerzo (usabilidad) de los usuarios para crear exactamente los mismos módulos en moodle, claroline, atutor y con la herramienta creada.

La reducción es del 67.4% en tiempo y 72.5% en esfuerzo, con estos resultados, los autores concluyen que el uso de la Ingeniería Dirigida por Modelos reduce significativamente el tiempo y esfuerzo en la construcción y despliegue de cursos sobre plataformas LMS.

3.3.5 Desarrollo de Aplicaciones de Redes Inalámbricas de Sensores

[Rodrigues+ 2011] desarrollaron una herramienta para facilitar el desarrollo de aplicaciones de redes inalámbricas de sensores. Los objetivos planteados son: (1) facilitar el desarrollo de aplicaciones de redes inalámbricas de sensores, (2) promover la separación se responsabilidades entre los diferentes expertos involucrados en la construcción de este tipo de aplicaciones y (3) promover el reuso de artefactos de software comprendidos en sistemas de redes inalámbricas de sensores.

El conocimiento representando diferentes plataformas de sensores es especificado en el nivel PSM. Por lo tanto la infraestructura MDA propuesta abarca diferentes metamodelos PSM, uno para cada plataforma WSN. Se usaron 2 plataformas de sensores: TinyOS y Sun SPOT. Todos los metamodelos fueron desarrollados utilizando el lenguaje de Ecore. La Figura 3-12 muestra el metamodelo para TinyOS.

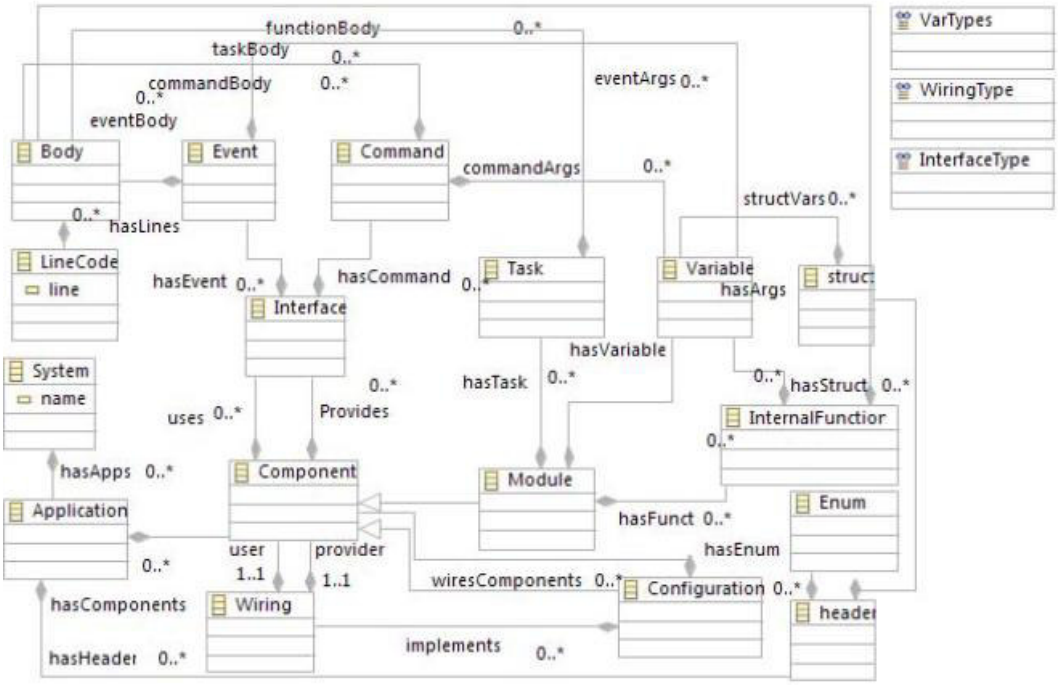


Figura 3-12 Metamodelo TinyOS [Rodrigues+ 2011]

En el proceso MDA propuesto, expertos del dominio de aplicación están a cargo del modelado del PIM, luego un primer conjunto de transformaciones semiautomáticas toma como entrada este PIM, junto con el metamodelo PSM de la plataforma del sensor elegida, y generan la instancia PSM que representa la aplicación lógica mapeada a la plataforma elegida.

Los expertos de redes incrementan los artefactos de software (la instancia PSM) incluyendo nuevos elementos o agregando detalles a los elementos existentes por lo que el PSM final es suficiente para describir completamente los requerimientos de la aplicación en la plataforma elegida.

En un siguiente paso, otro conjunto de transformaciones MDA toman como entrada el refinado modelo PSM y generan el código fuente ejecutable.

GenModel de EMF fue usado para crear automáticamente el editor del modelo desde los metamodelos. Para cada plataforma de sensor, existe un conjunto de transformaciones M2M (Model to Model) y M2T (Model to Text).

Se usó OMG QVTO (Operational QVT Language) para las transformaciones M2M.

Se usó el plugin de eclipse Acceleo para las transformaciones M2T. Acceleo es una implementación del lenguaje de transformación MOF Model to Text (MOFM2T) que es un actual estándar OMG.

Para generar el PSM y código fuente, transformaciones son ejecutadas dentro del entorno Eclipse. La Figura 3-13 muestra un diagrama con el comportamiento de la aplicación que se ejecuta en los nodos de sensores.

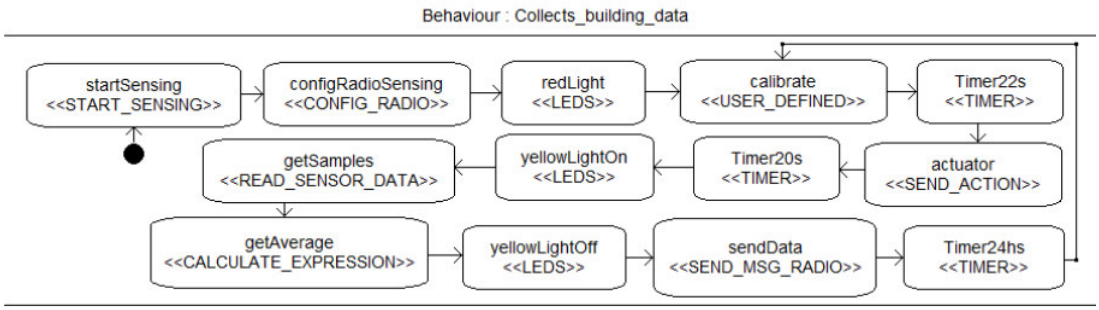


Figura 3-13 Diagrama del Comportamiento de la Aplicación Inalámbrica de Redes de Sensores [Rodrigues+ 2011]

En la evaluación se midieron las siguientes métricas: (1) porcentaje de líneas de código generadas, se generó el 100% del código. (2) número de errores en el código, el código generado está libre de errores, código correcto y funcional y (3) reducción del esfuerzo de desarrollo, desarrollo de las aplicaciones en mucho menos tiempo.

El experimento demuestra la reducción significativa en el esfuerzo de programación lograda con el enfoque MDA. Uso solo del 8.33% del tiempo requerido usando métodos tradicionales de programación manual.

3.3.6 Desarrollo de Sistemas Multi-Agente

[Gascueña+ 2012] implementan una herramienta para el desarrollo de sistemas multi-agente. La funcionalidad de la herramienta propuesta no es nueva. Esta propuesta es una innovación tecnológica para tomar ventaja de los beneficios de MDE. EMF es usado para especificar el metamodelo Ecore Prometheus (PEMM), necesario para describir los elementos de modelado de la metodología de desarrollo de sistemas multi-agente Prometheus. GMF es usado para construir el editor gráfico para especificar los modelos de Prometheus. El metamodelo de la metodología Prometheus es mostrado en la Figura 3-14.

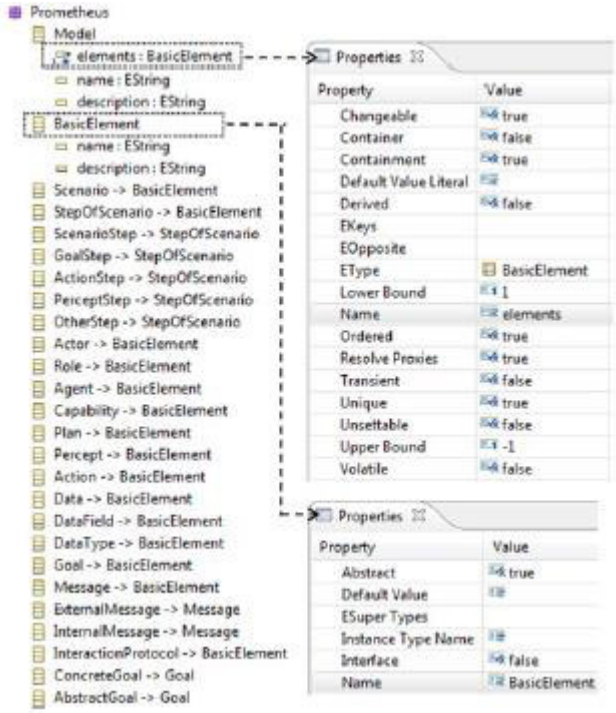


Figura 3-14 Metamodelo de la Metodología de Desarrollo de Sistemas Multi-Agentes Prometheus [Gascueña+ 2012]

El editor de modelos desarrollado es llevado a cabo siguiendo 3 actividades: (1) Se crea un metamodelo que especifique los conceptos de modelado de la metodología Prometheus, (2) Luego, GMF es usado para desarrollar el editor gráfico que soporte la creación de modelos gráficos, de acuerdo con el metamodelo Prometheus creado, (3) Se implementa la funcionalidad con el código escrito en JACK (Lenguaje de programación de agentes).

El editor gráfico es creado a través de la guía de proceso ofrecida por GMF, la cual nos permite crear un editor gráfico desde un modelo de dominio. Los modelos generados

son los siguientes: (1) Un modelo generador (.genmodel) es obtenido usando el metamodelo Ecore, específicamente un editor basado en árbol es creado para especificar modelos usando este generador de modelos (.genmodel). (2) Después un modelo gráfico (.gmfgraph) que define las figuras, nodos y enlaces en el diagrama es derivado. Este modelo especifica los elementos del modelo que tienen una representación gráfica, como nodo, relación o atributo. (3) Luego, el modelo de herramientas (.gmftool) que define la paleta del editor gráfico es derivado. Este modelo especifica los elementos que están en la paleta y como son mostrados (como relaciones o como nodos). (4) Seguido, enlaces entre el modelo del dominio (.ecore), el modelo gráfico (.gmfgraph) y el modelo de herramientas (.gmftool) son registrados en el modelo de mapeo (.gmfmap). (5) Finalmente, el modelo de mapeo (.gmfmap), junto con el modelo generador (.genmodel), es usado por otro wizard para obtener un modelo generador gráfico GMF. (6) El modelo generador gráfico (.gmfgen) es empleado por GMF de eclipse para automáticamente generar el código Java el editor gráfico.

Una vez el código generado es compilado como un nuevo plugin de eclipse, este puede ser usado para crear nuevos modelos Prometheus. JET es usado en el editor para proveer habilidades de generación de código.

Los autores concluyen que MDE mejora la calidad de los sistemas desarrollados en términos de productividad, portabilidad, interoperabilidad y mantenimiento. La interfaz de la herramienta se muestra en la Figura 3-15.

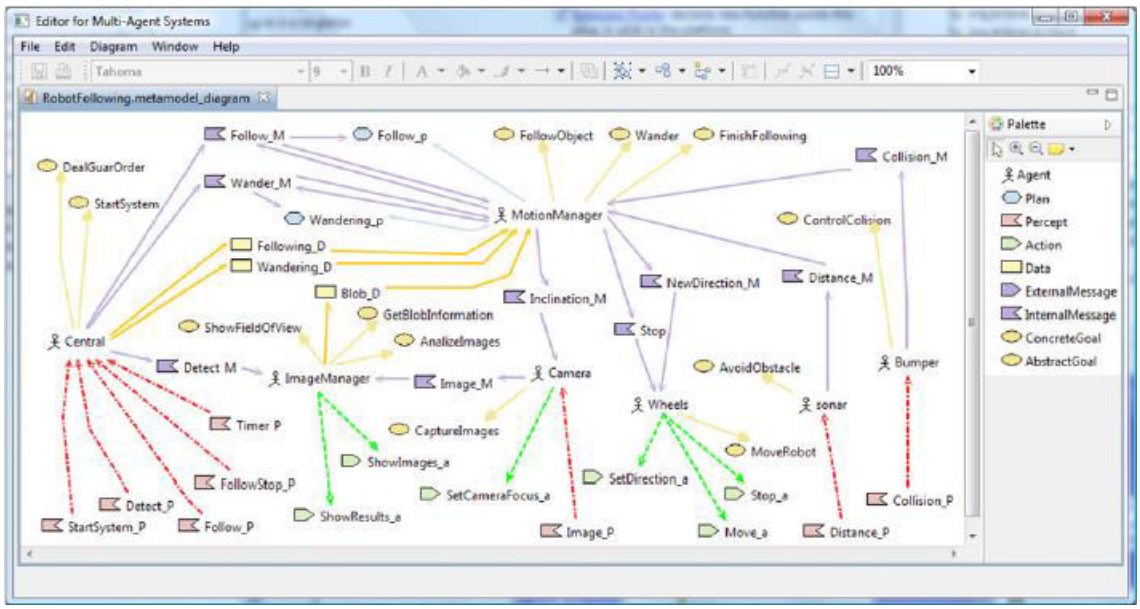


Figura 3-15 Interfaz de la Herramienta de Modelado de Sistemas Multi-Agente [Gascueña+ 2012]

3.3.7 Desarrollo de Sistemas Basado en Componentes

[Lau+ 2012] construyeron una herramienta para soportar el desarrollo basado en componentes de acuerdo a un modelo de componentes que también propusieron. Esta herramienta está implementada basa en MDE usando Generic Modeling Environment (GME).

GME es un entorno de modelado genérico y personalizable. Es comparable al estándar GMF, la personalización de GME se inicia con el metamodelado. Un metamodelo en GME es esencialmente un diagrama de clases con conceptos bien conocidos como herencia y contención.

Se definieron 2 metamodelos, uno para el ciclo de vida del componente y otro para el sistema. La Figura 3-16 muestra el metamodelo para el ciclo de vida del componente.

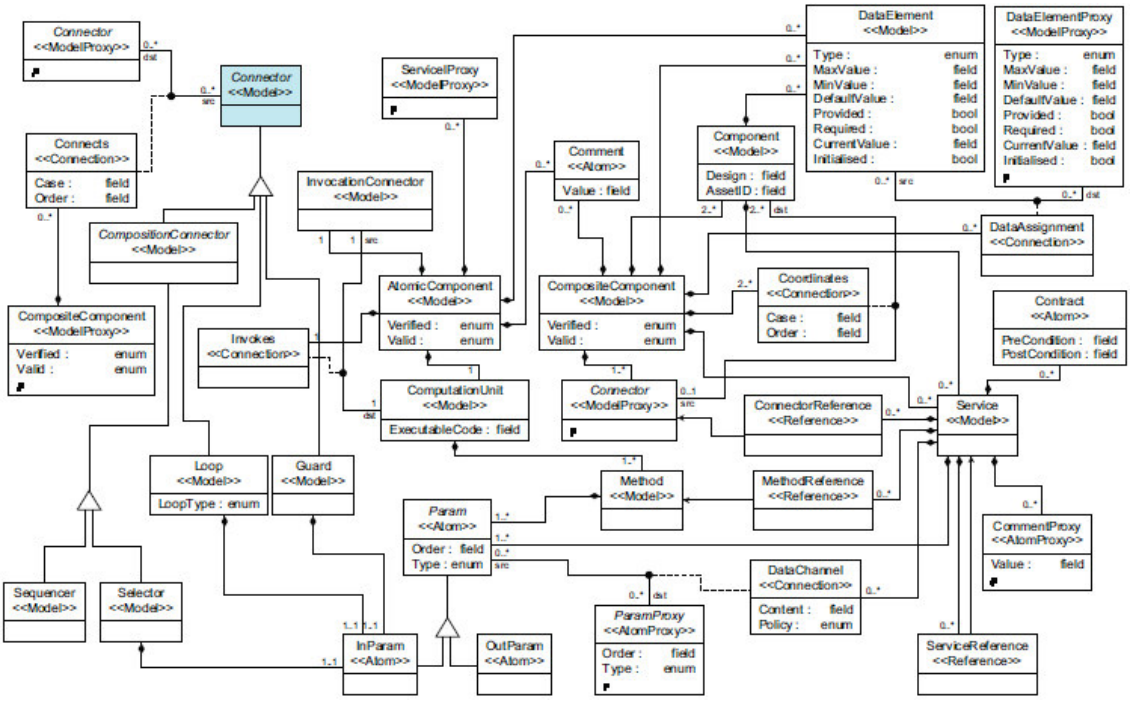


Figura 3-16 Metamodelo del Ciclo de Vida del Componente [Lau+ 2012]

Hay nociones de proxy y referencia. Proxy permite clonar un elemento del metamodelo por lo que este elemento clonado pueda coexistir y ser organizado convenientemente en un extenso y complejo metamodelo. Esto permite definir diagramas claros y bien definidos. Referencia permite referencias entre elementos del metamodelo.

Una vez que el metamodelo es creado, necesita ser registrado con GME. Luego de esto modelos pueden ser instanciados desde el metamodelo registrado.

Una importante característica de GME es el concepto de aspecto. Un aspecto define una vista en la cual elementos del metamodelo especificado son mostrados. En general podría haber superposición entre los aspectos si los elementos del metamodelo compartido se utilizan para enlazar aspectos semánticamente. Aspectos son parte del metamodelo en GME y por la tanto serán registrados a GME como parte del registro del metamodelo.

Intérprete es otro importante concepto en GME. Un intérprete puede tomar un modelo activo y procesarlo. Procesamiento de modelos puede variar al insertar y modificar elementos del modelo. Traversing un modelo es quizá lo más interesante ya que nos habilita realizar la validación, transformación y ejecución del modelo.

Un intérprete necesita una implementación concreta para determinar cómo la interpretación del modelo es hecha. Para definir un intérprete, GME provee un wizard embebido en Microsoft Visual Studio. GME genera clases C++ para elementos del metamodelo.

Se usó C++ para implementar todos los intérpretes. Se necesita asegurar que los modelos no son contradictorios, por tanto se implementó una herramienta de validación llamada VAL. La Figura 3-17 muestra la interfaz de usuario de la herramienta.

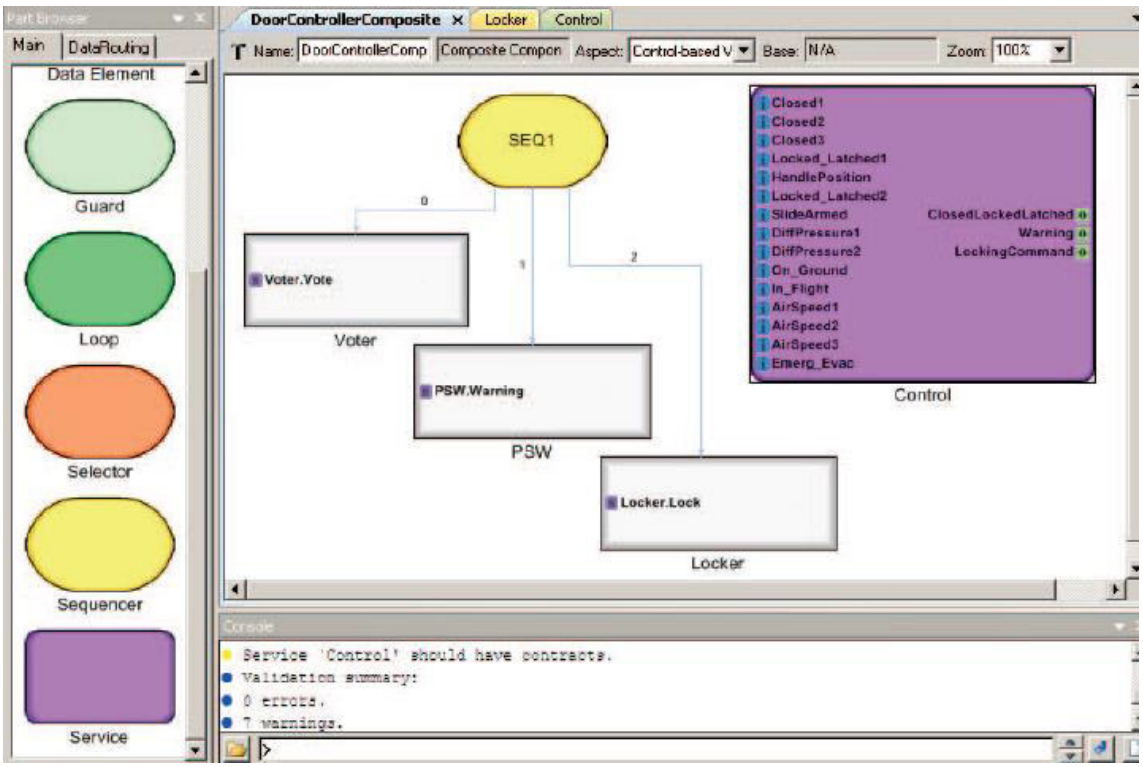


Figura 3-17 Interfaz de la Herramienta para el Desarrollo Basado en Componentes [Lau+ 2012]

3.3.8 Modelado de Puntos de Vista de Arquitectura de Software

[Tekinerdogan+ 2013] propone una herramienta llamada SAVE-BENCH para modelar puntos de vista de arquitectura de software como DSLs.

Cada definidor de un punto de vista crea la definición de la gramática del punto de vista usando el editor Xtext.

Xtext es parte del proyecto Eclipse TMF (Textual Modeling Framework) y permite la creación de DSLs para definiciones de gramáticas.

Después de escribir la gramática, el lenguaje generador Xtext es ejecutado la cual construye la implementación total del DSL para la gramática escrita.

Luego el generador Xtext extrae el metamodelo de la gramática y lo retorna como un metamodelo Ecore.

Los metamodelos desarrollados se muestran en la Figura 3-18.

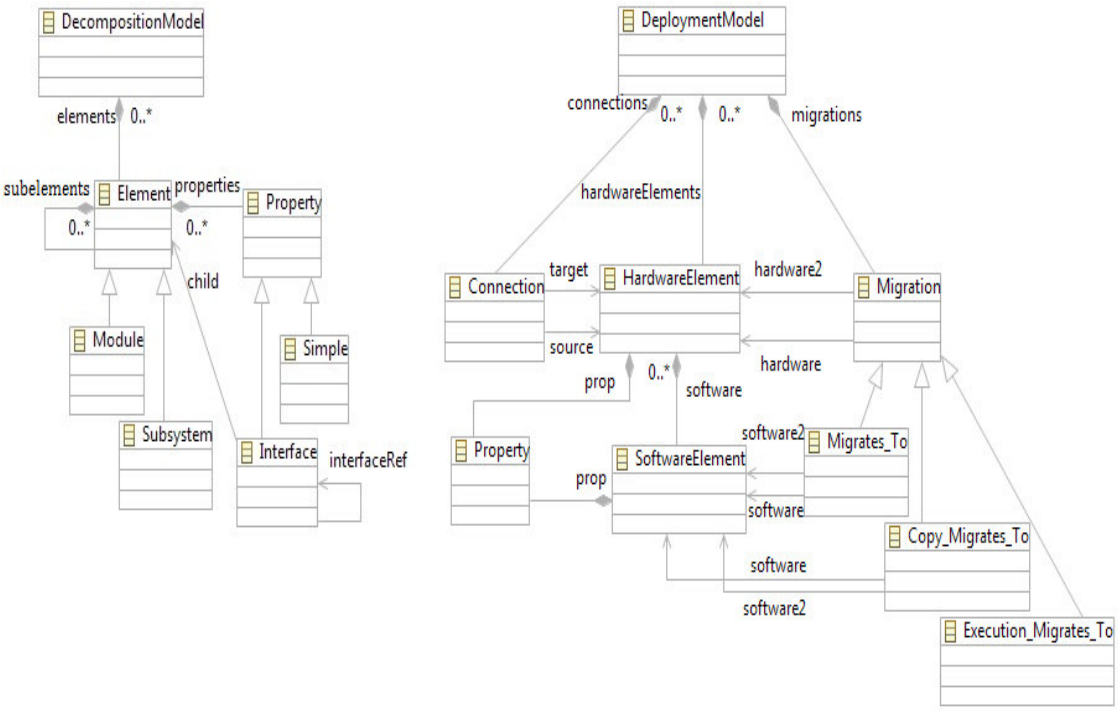


Figura 3-18 Metamodelos para el Modelado de Puntos de Vista de Arquitectura de Software [Tekinerdogan+ 2013]

Se usa este metamodelo ecore como la definición de la sintaxis abstracta mientras se define la sintaxis concreta visual del correspondiente DSL.

Para soportar un desarrollo fácil los autores usaron la herramienta Eugenia para generar los modelos necesarios para la generación de diagramas GMF desde un metamodelo Ecore con anotaciones.

Para anotar el metamodelo ecore con información de sintaxis concreta visual, se ha utilizado EMFatic. Esto es, usando anotaciones específicas el definidor de puntos de vistas establece para cada elemento del metamodelo (punto de vista) las correspondientes notaciones gráficas.

El resultante metamodelo Ecore es dado como input al generador Eugenia, la cual genera los modelos requeridos para la generación del editor gráfico GMF.

Luego tanto el editor textual y visual definido para el punto de vista son exportados como plugins para eclipse.

Un modelador de vistas puede usar estos editores para modelar las vistas de arquitectura basada en el punto de vista.

La interfaz de usuario de la herramienta es mostrada en La Figura 3-19.

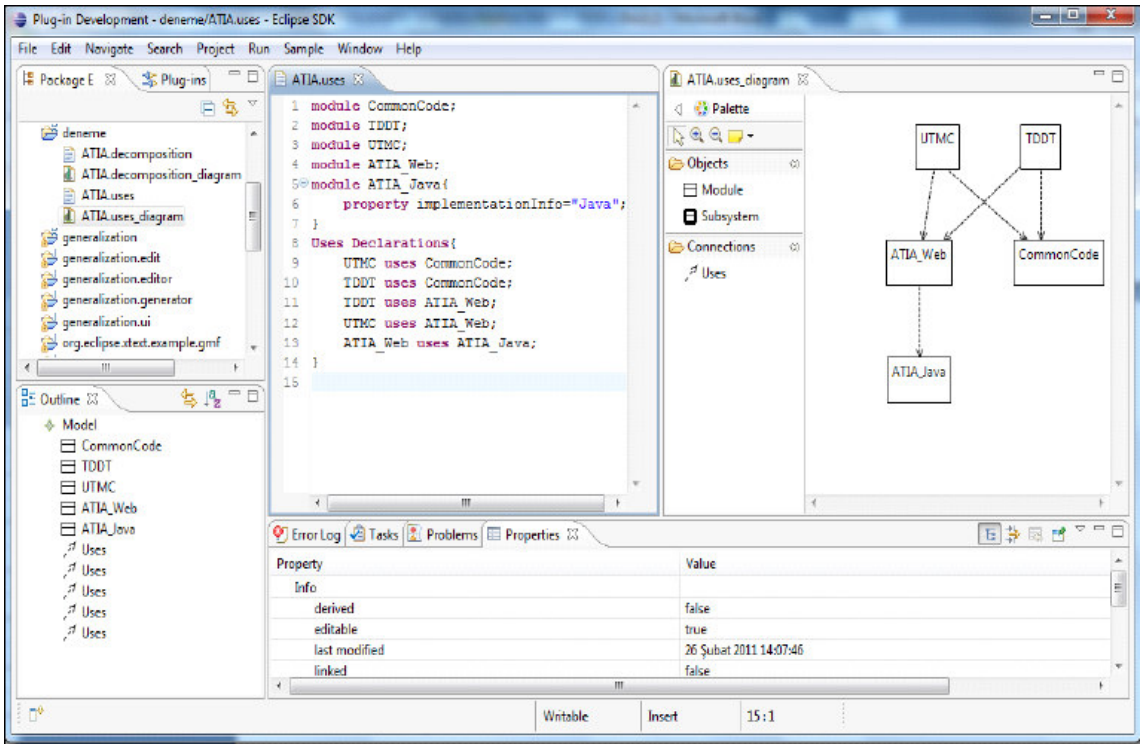


Figura 3-19 Interfaz de la Herramienta para Modelar Vistas de Arquitectura de Software [Tekinerdogan+ 2013]

3.4 Comparación de Frameworks y Herramientas MDE

Los autores de los artículos revisados en el estado del arte han usado diferentes frameworks y herramientas para la creación de un editor gráfico de modelado (DSM) basado en la Ingeniería Dirigida por Modelos (MDE). La Tabla 3-1 muestra la comparación de estos frameworks y herramientas.

Características	**Frameworks/Herramientas**					
	GMF	**Kybele GMFGen**	**Eugenia**	**GEMS**	**VSVMSDK Antes DSL Tools**	**GME**
Generación Automática del Editor Gráfico	No	Sí	Sí	Sí	No	No
Software Libre	Sí	Sí	Sí	Sí	No	Sí
Multiplataforma	Sí	Sí	Sí	Sí	No	No
Documentación y Soporte	Sí	No	Sí	Sí	Sí	Sí
Tecnología Madura	Sí	No	Sí	No	Sí	Sí

Tabla 3-1 Tabla Comparativa de Frameworks y Herramientas de Soporte a la Ingeniería Dirigida por Modelos [Elaboración Propia]

En la comparación mostrada en la Tabla 3-1, se puede observar que se seleccionó la herramienta Eugenia [Kolovos+ 2010] del proyecto Epsilon [Epsilon 2013] de Eclipse para el desarrollo del editor gráfico de la herramienta propuesta. Eugenia está basada en el framework Graphical Modeling Framework (GMF) de la plataforma Eclipse. La plataforma Eclipse fue elegida porque es una tecnología madura, software libre y multiplataforma. Se eligió usar el proyecto Epsilon dado que es una familia completa de lenguajes y herramientas maduras para desarrollar software basado en MDE. La base de Epsilon es el lenguaje Epsilon Object Language (EOL). Específicamente para la validación de modelos de replicación MySQL se eligió al lenguaje Epsilon Validation Language (EVL) y para la generación de los comandos mysqlreplicate de configuración se eligió al lenguaje Epsilon Generation Language (EGL).

3.5 Resumen del Capítulo

En este capítulo se ha descrito el estado del arte relacionado con esta investigación. Este capítulo se organizó en cuatro secciones.

En la primera sección del capítulo se describe en modo general la revisión de la literatura realizada en esta investigación.

En la siguiente sección se describen tres herramientas de diagramación para el modelado de la replicación de MySQL.

En la tercera sección se describe el desarrollo de ocho herramientas basadas en la Ingeniería Dirigida por Modelos (MDE). Los autores de estos artículos detallan el proceso a seguir para el desarrollo de una herramienta basada en MDE utilizando diferentes frameworks y herramientas para MDE.

En la última sección se realiza la comparación de los frameworks y herramientas para el desarrollo de un editor gráfico basado en MDE y se indican los seleccionados para el desarrollo de la herramienta propuesta.

En el siguiente capítulo se detalla el proceso de desarrollo de la herramienta propuesta en esta investigación.

Capítulo 4: Desarrollo de la Herramienta

En este capítulo se detalla el proceso de desarrollo de la herramienta propuesta "MySQL Replication Modeling" siguiendo las fases del Proceso Unificado de Rational (RUP) y basada en la Ingeniería Dirigida por Modelos (MDE). En el siguiente capítulo se describe la evaluación de la herramienta desarrollada.

4.1 Proceso de Desarrollo de la Herramienta

Para el proceso de desarrollo de la herramienta se siguieron las fases del Proceso Unificado de Rational (RUP) [Kruchten 2000]. RUP es un producto de Rational (IBM) que se caracteriza por ser iterativo e incremental, estar centrado en la arquitectura y guiado por los casos de uso.

RUP divide el proceso en cuatro fases, dentro de las cuales se realizan varias iteraciones en número variable según la complejidad del software y en las que se hace un mayor o menor hincapié en las distintas disciplinas y actividades [Kruchten 2000].

RUP fue adaptado y simplificado en esta investigación para un desarrollo basado en la Ingeniería Dirigida por Modelos (MDE), tal como se puede observar en la Tabla 4-1 donde se muestran las disciplinas trabajadas por cada fase del RUP en el desarrollo de la herramienta.

Fase	Disciplina
1. Inicio	Análisis del Dominio
	Requerimientos
2. Elaboración	Metamodelo del Dominio
	Análisis y Diseño
3. Construcción	Implementación
	Pruebas
4. Transición	Despliegue

Tabla 4-1 Fases y Disciplinas del Proceso de Desarrollo de la Herramienta [Elaboración Propia]

A continuación se describe cada una de las fases y disciplinas del proceso de desarrollo.

4.2 Fase de Inicio

Esta es la primera fase que define el RUP, aquí se ha trabajado en las disciplinas de análisis del dominio y de requerimientos. A continuación se describe cada una de estas disciplinas.

4.2.1 Disciplina de Análisis del Dominio

Esta es la disciplina base para poder cumplir con la funcionalidad de la herramienta propuesta. En esta disciplina se realizó lo siguiente:

- Estudiar la documentación oficial de la replicación de MySQL.
- Analizar detalladamente las topologías soportadas en la replicación MySQL.
- Identificar las consideraciones a tener en cuenta para configurar la replicación en MySQL.
- Configuración práctica de la replicación en MySQL ejecutando el comando **mysqlreplicate** en la herramienta MySQL Utilities.
- Consultar con los expertos del dominio.

En la replicación de MySQL, a un servidor principal se le conoce como maestro, y a una réplica se le denomina esclavo [MySQL 2013b], por lo consiguiente la replicación en MySQL permite duplicar los datos desde un servidor maestro hacia uno o más servidores esclavos.

Las consideraciones a tener en cuenta para la validación de modelos de replicación en MySQL son las siguientes [MySQL 2013e]:

- Un servidor esclavo puede tener uno y solo uno como servidor maestro.
- Un servidor maestro puede tener cualquier número de esclavos.
- Un servidor no puede tener una relación de replicación consigo mismo.
- La unión del host y puerto de una instancia de MySQL debe ser única.
- Cada servidor debe tener un identificador numérico único establecido en el parámetro de configuración **server-id**. El rango de valores de este identificador está entre 1 y 2^{32}.

MySQL soporta las topologías de replicación mostradas en la Figura 4-1.

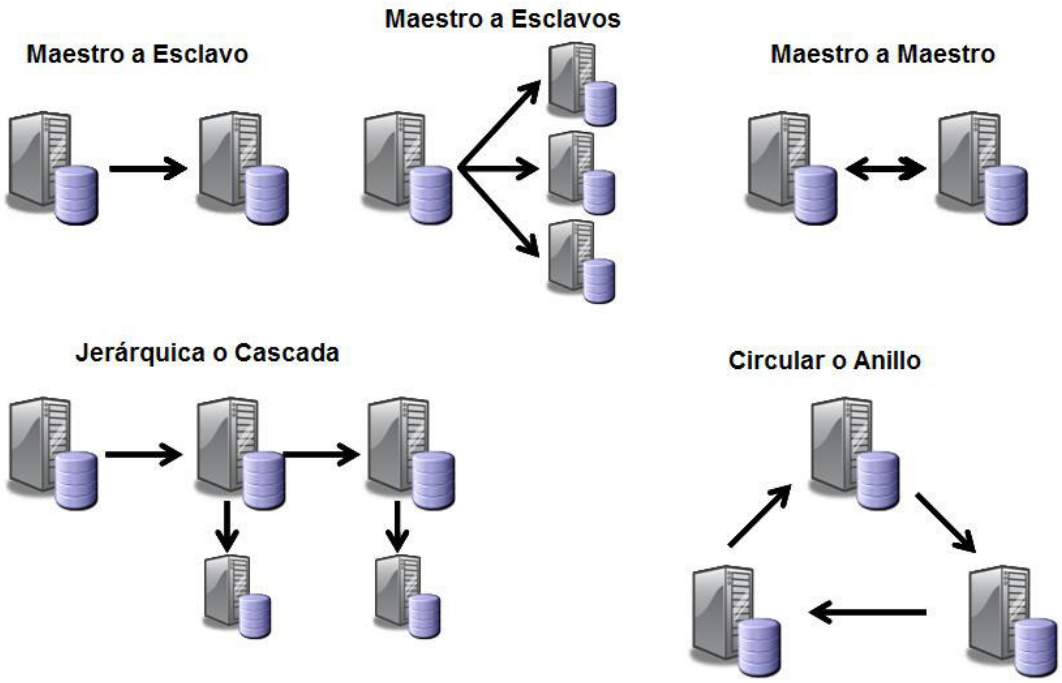


Figura 4-1 Topologías de Replicación en MySQL [MySQL 2013e]

La configuración de la replicación de MySQL entre un servidor maestro y un servidor esclavo es una tarea sencilla, para lograr ello, un administrador de base de datos (DBA) o desarrollador en MySQL debe realizar lo siguiente:

- Establecer los parámetros relacionados con la replicación en el archivo de configuración tanto del servidor maestro como del servidor esclavo de MySQL. Luego de modificar el archivo de configuración de cada servidor, se debe de reiniciar el servicio de MySQL para que los cambios surtan efecto.
- Escribir y ejecutar el comando **mysqlreplicate** en la herramienta MySQL Utilities indicando como parámetros los datos del servidor maestro y esclavo.

La Figura 4-2 muestra la sintaxis y ejecución del comando **mysqlreplicate** para configurar la replicación entre un servidor maestro y un servidor esclavo.

```
mysqlreplicate --master=root@black:3306 --slave=root@blue:3306
# master on black: ... connected.
# slave on blue: ... connected.
# Checking for binary logging on master...
# Setting up replication...
# ...done.
```

Figura 4-2 Sintaxis del Comando mysqlreplicate

A continuación se describe la disciplina de requerimientos.

4.2.2 Disciplina de Requerimientos

En esta disciplina se describen los requerimientos funcionales y no funcionales de la herramienta propuesta MySQL Replication Modeling.

4.2.2.1 Requerimientos Funcionales

Los requerimientos funcionales fueron capturados mediante la técnica de casos de uso

En la Tabla 4-2 se describen los casos de uso para modelar la replicación de MySQL.

#	Caso de Uso	Descripción
CU-01	Crear Modelo de Replicación	La herramienta permitirá crear un modelo de replicación entre servidores MySQL.
CU-02	Guardar Modelo de Replicación	La herramienta permitirá guardar un modelo de replicación de MySQL en formato XMI (XML Metadata Interchange).
CU-03	Abrir Modelo de Replicación	La herramienta permitirá abrir un modelo de replicación entre servidores MySQL.
CU-04	Agregar Servidor MySQL	La herramienta permitirá agregar un servidor MySQL a un modelo de replicación MySQL.
CU-05	Modificar Servidor MySQL	La herramienta permitirá modificar los datos de un servidor MySQL de un modelo de replicación MySQL.
CU-06	Eliminar Servidor MySQL	La herramienta permitirá eliminar un servidor MySQL de un modelo de replicación MySQL.
CU-07	Crear Relación Maestro-Esclavo	La herramienta permitirá crear una relación maestro-esclavo en un modelo de replicación MySQL.
CU-08	Eliminar Relación Maestro-Esclavo	La herramienta permitirá eliminar una relación maestro-esclavo de un modelo de replicación MySQL.
CU-09	Crear Relación Maestro-Maestro	La herramienta permitirá crear una relación maestro-maestro en un modelo de replicación MySQL.
CU-10	Eliminar Relación Maestro-Maestro	La herramienta permitirá eliminar una relación maestro-maestro de un modelo de replicación MySQL.

Tabla 4-2 Descripción de Casos de Uso para Modelar la Replicación de MySQL [Elaboración Propia]

En la Tabla 4-3 se describen los casos de uso para validar un modelo de replicación de MySQL, así como para generar los comandos mysqlreplicate de configuración a partir de un modelo de replicación de MySQL.

#	Caso de Uso	Descripción
CU-11	Validar Modelo de Replicación	La herramienta permitirá validar si un modelo de replicación MySQL contiene o no errores, listando los errores en caso existan.
CU-12	Generar Comandos de Configuración	La herramienta permitirá generar automáticamente los comandos mysqlreplicate de configuración a partir de un modelo de replicación MySQL.

Tabla 4-3 Descripción de Casos de Uso para la Validación y Generación de Comandos de un Modelo de Replicación de MySQL [Elaboración Propia]

A continuación se describen los requerimientos no funcionales.

4.2.2.2 Requerimientos No Funcionales

Los requerimientos no funcionales de la herramienta de describen en la Tabla 4-4.

#	Requerimiento No Funcional	Descripción
RNF-01	Usabilidad	La herramienta debe tener una interfaz fácil de usar. Todos los elementos del modelo de replicación MySQL deben ser fácilmente accesibles por el usuario. El idioma de todos los elementos de la herramienta debe estar en inglés.
RNF-02	Escalabilidad	La herramienta debe tener la capacidad de trabajar sin problemas un modelo de replicación con 200 servidores MySQL.
RNF-03	Integración	La herramienta será integrada con el IDE de Eclipse mediante plugins.
RNF-04	Portabilidad	La herramienta debe poder ejecutarse en sistemas operativos Windows, Linux y MAC.

Tabla 4-4 Requerimientos No Funcionales [Elaboración Propia]

4.3 Fase de Elaboración

En esta fase se describe la disciplina de metamodelo del dominio y de análisis y diseño.

4.3.1 Disciplina de Metamodelo del Dominio

El metamodelo del dominio define la sintaxis abstracta de un DSL. El metamodelo es un artefacto de vital importancia en un desarrollo MDE ya que es el dato de entrada para la generación del editor gráfico (sintaxis concreta) de un DSL. Para su definición se necesita realizar como requisitos previos el análisis del dominio y la especificación de los requerimientos. La Figura 4-3 muestra el metamodelo propuesto.

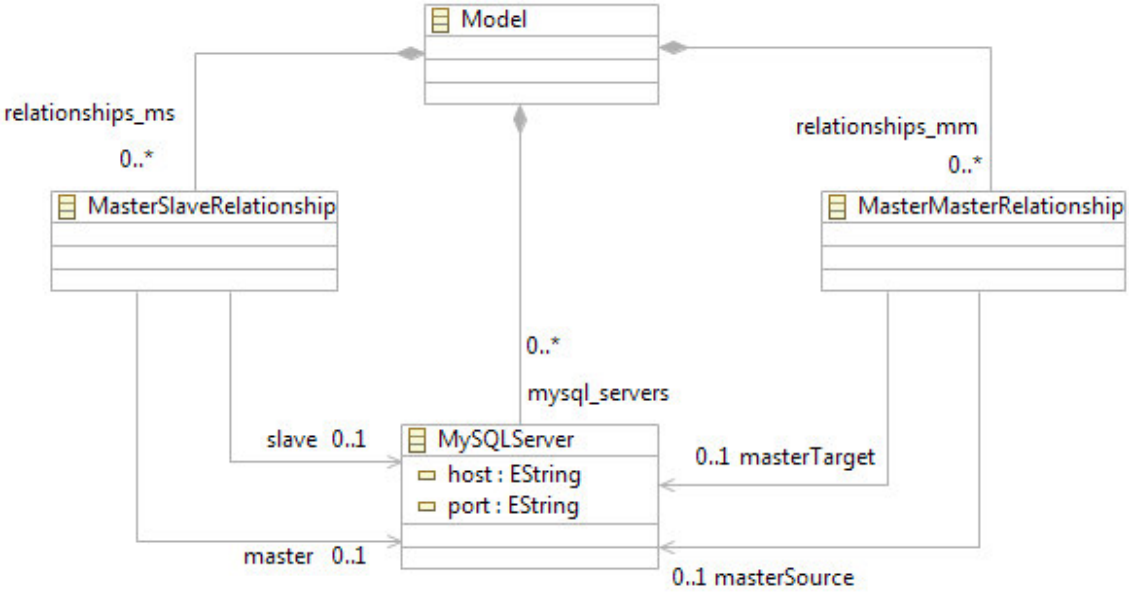


Figura 4-3 Metamodelo Propuesto [Elaboración Propia]

A continuación se describe cada uno de los elementos del metamodelo.

- **Model**: Representa el elemento raíz, que viene a ser el diagrama que contendrá todo el modelo y es definida para tener nodos (servidores MySQL) y enlaces entre nodos (relaciones maestro-esclavo y maestro-maestro entre servidores).
- **MySQLServer**: Representa a los servidores MySQL que son parte de un modelo de replicación de MySQL.
- **MasterSlaveRelationship**: Representa a una relación maestro-esclavo entre un par de servidores MySQL.
- **MasterMasterRelationship:** Representa a una relación maestro-maestro entre un par de servidores MySQL.

4.3.2 Disciplina de Análisis y Diseño

En esta disciplina se trabajó en base a las vistas arquitectónicas del documento de arquitectura de software (Software Architecture Document, SAD). La intención es capturar y comunicar las decisiones arquitectónicas significativas que se han realizado sobre la herramienta. Concretamente se mostrará el diseño de la arquitectura a través de las diferentes vistas que muestran distintos aspectos del sistema como son: Vista de Casos de Uso, Vista Lógica, Vista de Implementación, Vista de Despliegue y Vista de Datos, cada una de las cuales ilustra un aspecto en particular del software desarrollado. Se pretende de esta forma brindar una visión global y comprensible del diseño general de la arquitectura desarrollada. A continuación se describe cada una de las vistas.

4.3.2.1 Vista de Casos de Uso

Esta vista muestra la funcionalidad de la herramienta. La Figura 4-4 muestra el modelo de casos de uso para el modelado de la replicación de MySQL.

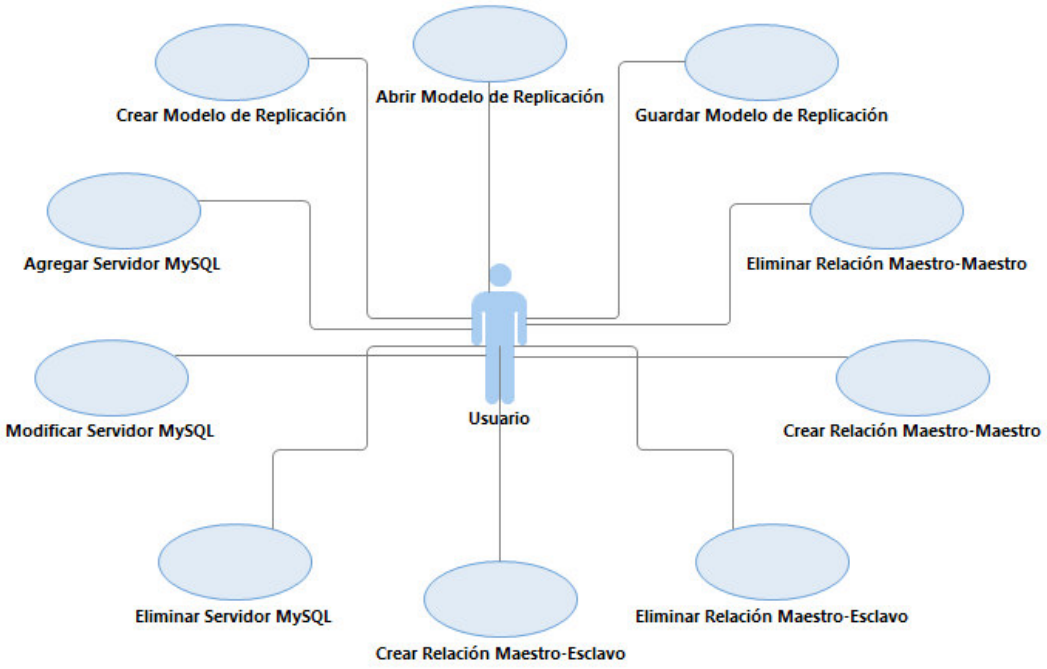


Figura 4-4 Modelo de Casos de Uso para el Modelado de la Replicación de MySQL [Elaboración Propia]

La Figura 4-5 muestra el modelo de casos de uso para validar un modelo de replicación de MySQL, así como para generar los comandos mysqlreplicate de configuración a partir de un modelo de replicación de MySQL.

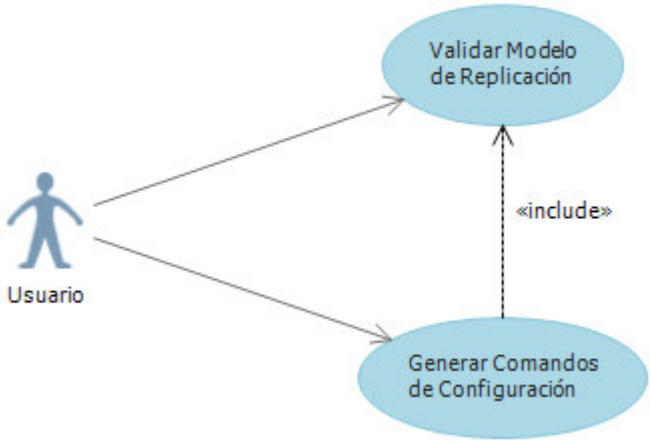


Figura 4-5 Modelo de Casos de Uso para la Validación y Generación de Comandos de Configuración de un Modelo de Replicación de MySQL [Elaboración Propia]

La Tabla 4-5 muestra la especificación del caso de uso Validar Modelo de Replicación.

CU-11	**Validar Modelo de Replicación**	
Descripción	La herramienta deberá comportarse tal como se describe el siguiente caso de uso para validar un modelo de replicación de MySQL.	
Pre-Condición	No Aplica	
Flujo Normal	**Paso**	**Acción**
	1	El usuario de la herramienta da inicio al caso de uso Validar Modelo de Replicación.
	2	La herramienta verifica los errores del modelo de replicación.
	3	Si el modelo contiene errores, la herramienta (i) muestra los errores del modelo en una vista de problemas, (ii) muestra el número de servidores y relaciones de replicación del modelo, así como el tiempo tomado en la validación en una vista de consola, y finalmente (iii) resalta con un icono rojo los servidores y relaciones de replicación que presentan errores.
Post-Condición	No Aplica	
Excepciones	**Paso**	**Acción**
	1	Si la herramienta determina que no existe abierto un modelo de replicación, lo informará al usuario mostrando un mensaje en la vista de consola.
	2	Si la herramienta determina que el modelo de replicación no presenta errores, lo informará al usuario mostrando un mensaje en la vista de consola.

Tabla 4-5 Especificación del Caso de Uso Validar Modelo de Replicación [Elaboración Propia]

La especificación del caso de uso Generar Comandos de Configuración se muestra en la Tabla 4-6.

CU-12	Generar Comandos de Configuración	
Descripción	La herramienta deberá comportarse tal como se describe el siguiente caso de uso para generar los comandos de configuración a partir de un modelo de replicación de MySQL.	
Pre-Condición	No Aplica	
Flujo Normal	**Paso**	**Acción**
	1	El usuario de la herramienta da inicio al caso de uso Generar Comandos de Configuración.
	2	La herramienta invoca al caso de uso Validar Modelo de Replicación.
	3	Si el modelo no contiene errores, la herramienta crea un archivo de texto dentro del proyecto del modelo con el contenido de los comandos de configuración del modelo de replicación de MySQL e informa al usuario con un mensaje de generación exitosa en la vista de consola.
Post-Condición	No Aplica	
Excepciones	**Paso**	**Acción**
	1	Si la herramienta determina que no existe abierto un modelo de replicación, lo informará al usuario mostrando un mensaje en la vista de consola.

Tabla 4-6 Especificación del Caso de Uso Generar Comandos de Configuración [Elaboración Propia]

4.3.2.2 Vista Lógica

Esta vista muestra la organización de la herramienta en dos subsistemas, uno para diagramar un modelo de replicación de MySQL llamado Editor Gráfico GMF y otro para la validación de un modelo de replicación y generación de comandos llamado personalizaciones, tal como se puede observar en la Figura 4-6.

Figura 4-6 Subsistemas de la Herramienta [Elaboración Propia]

Esta vista muestra también cómo la funcionalidad es diseñada en el interior del sistema. La Figura 4-7 muestra los componentes al interior de la herramienta MySQL Replication Modeling, así como los componentes externos que interactúan con la herramienta.

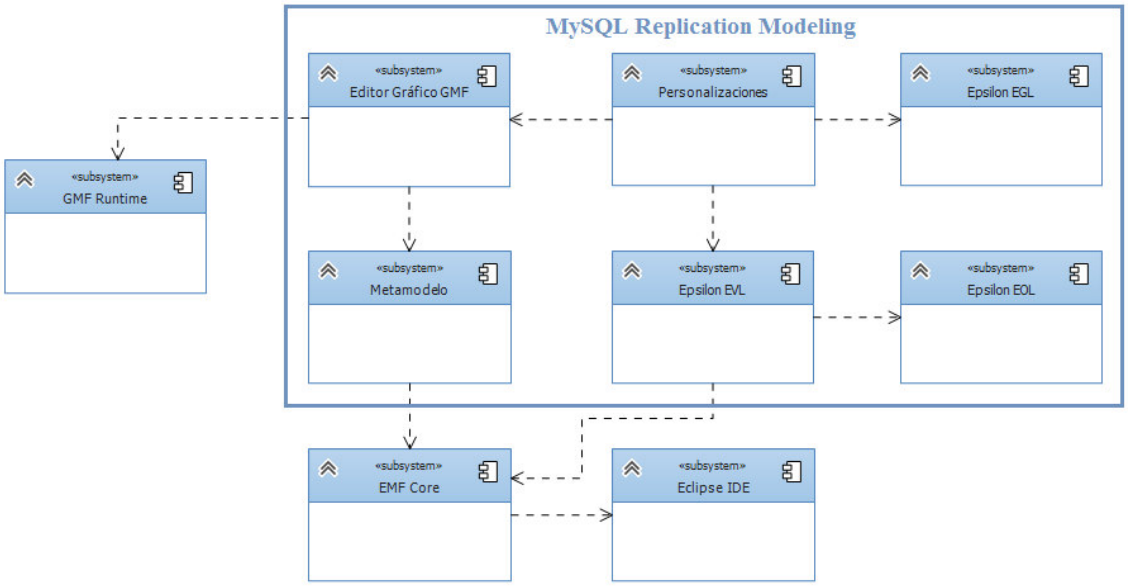


Figura 4-7 Arquitectura de Componentes de la Herramienta [Elaboración Propia]

A continuación se describe cada uno de los componentes de la arquitectura.

- ✓ **GMF Runtime:** Este componente simplifica la creación de diagramas proveyendo componentes reusables y mecanismos extensibles para mejorar el comportamiento de los diagramas en tiempo de ejecución. La herramienta hereda toda la funcionalidad de un editor gráfico GMF. Este componente provee el plugin .diagram, la cual es usada por la herramienta para manipular el comportamiento de los diagramas.
- ✓ **Editor Gráfico GMF:** Este componente contiene la base de la funcionalidad de la herramienta, tal como crear, mover, eliminar y editar nodos (servidores MySQL) y conexiones (relaciones maestro-esclavo y maestro-maestro) de un diagrama. Este componente está basado en el componente GMF Runtime de la cual hereda su funcionalidad básica. Este componente será generado por la herramienta Eugenia en base al componente del metamodelo.
- ✓ **Personalizaciones:** Este componente contiene la funcionalidad para personalizar al editor gráfico GMF. Este componente hace uso de los lenguajes

Epsilon Object Language (EOL), Epsilon Validación Language (EVL) y Epsilon Generation Language (EGL).

- ✓ **Epsilon Validation Language (EVL):** Para desarrollar las validaciones a los modelos de replicación de MySQL.
- ✓ **Epsilon Generation language (EGL): P**ara la generación de código tanto de los comandos mysqlreplicate de configuración de la replicación de MySQL.
- ✓ **Epsilon Object Language (EOL):** Usado para personalizar la estética gráfica de la herramienta.
- ✓ **Metamodelo:** Este componente es la sintaxis abstracta de un lenguaje de dominio específico (DSL). Conceptos abstractos tales como diagrama, nodos y conexiones son usados. Este componente es la implementación del metamodelo propuesto para modelar la replicación entre servidores MySQL.

  Este componente está basado en el componente EMF Core que provee el soporte de ejecución para los modelos y persistencia con serialización XML Metadata Interchange (XMI).
- ✓ **EMF Core:** Es uno de los componentes que comprende Eclipse Modeling Framework (EMF). Este componente contiene el metamodelo Ecore que será usado para describir el metamodelo de la herramienta propuesta. EMF Core es usado para generar el código Java (interfaces y clases de implementación) y proveer el soporte de ejecución para la herramienta incluyendo soporte para la persistencia de los diagramas en el formato estándar XML Metadata Interchange (XMI).
- ✓ **Eclipse IDE:** Este componente provee el soporte para todos los demás componentes antes descritos dado que la herramienta desarrollada consiste en una serie de plugins que se integrarán al IDE Eclipse.

El código fuente base de la herramienta será generado automáticamente por la herramienta Eugenia, tomando como entrada el metamodelo propuesto. La herramienta Eugenia generará un gran número de clases relacionado principalmente con los frameworks EMF, GEF y GMF.

Dado que en este momento se desconoce el gran número de clases que serán generadas, se modelará como diagrama de clases el metamodelo propuesto para el modelado de la replicación de MySQL.

La Figura 4-8 muestra el diagrama de clases mapeado con el metamodelo propuesto que luego será implementado textualmente en metaclases del metamodelo con el plugin Emfatic de Eclipse.

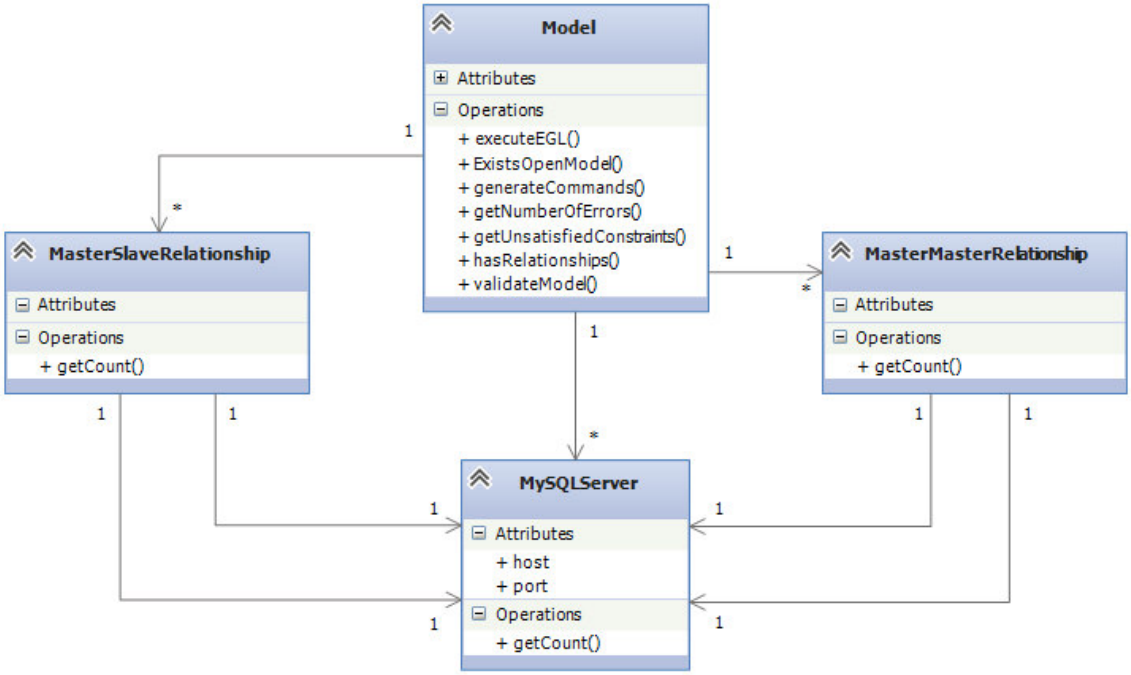


Figura 4-8 Diagrama de Clases del Metamodelo [Elaboración Propia]

- **Model**: Clase que representa un modelo de replicación de MySQL que contendrá los servidores MySQL y las relaciones maestro-esclavo y relaciones maestro-maestro entre los servidores MySQL.
- **MySQLServer**: Clase que representa a los servidores MySQL que son parte de un modelo de replicación de MySQL.
- **MasterSlaveRelationship**: Clase que representa a una relación maestro-esclavo entre un par de servidores MySQL.
- **MasterMasterRelationship**: Clase que representa a una relación maestro-maestro entre un par de servidores MySQL.

La Figura 4-9 muestra el diagrama de secuencia para el caso de uso Validar Modelo de Replicación y el diagrama de secuencia para el caso de uso Generar Comandos de Configuración se muestra en la Figura 4-10.

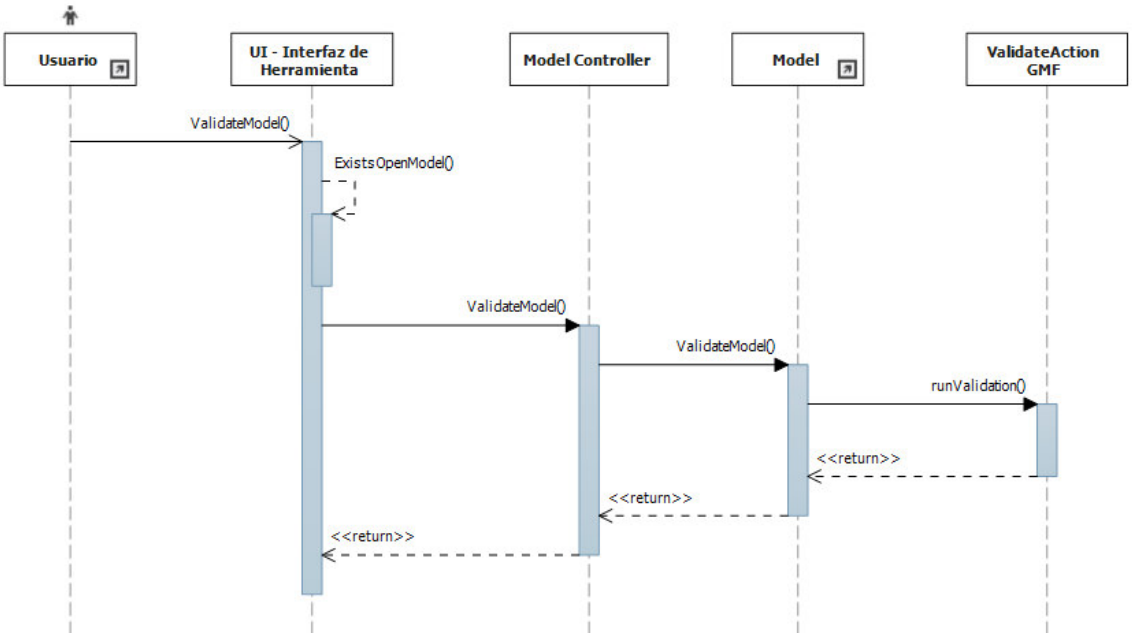


Figura 4-9 Diagrama de Secuencia del Caso de Uso Validar Modelo de Replicación [Elaboración Propia]

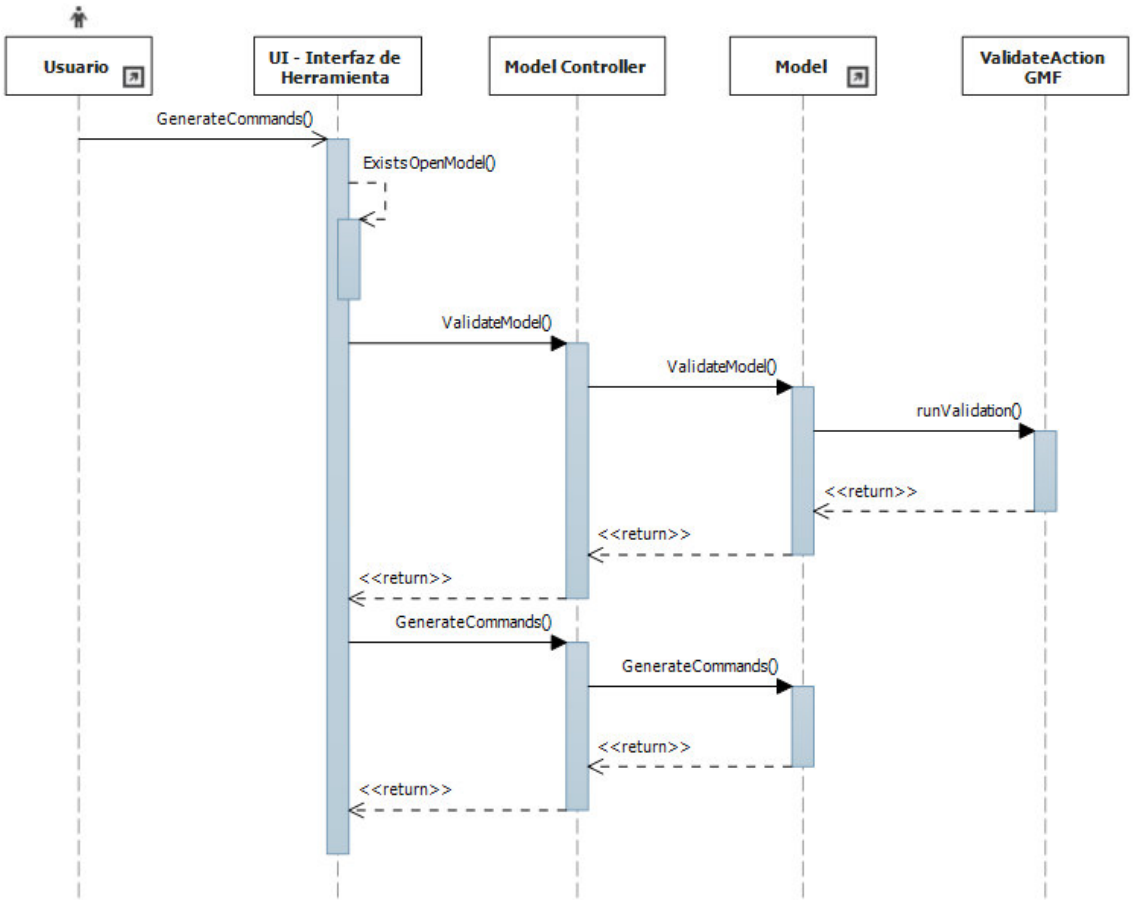


Figura 4-10 Diagrama de Secuencia del Caso de Uso Generar Comandos de Configuración [Elaboración Propia]

4.3.2.3 Vista de Implementación

Esta vista muestra la organización del código en un diagrama de componentes. El código fuente será generado automáticamente por la herramienta Eugenia a partir del metamodelo. La Figura 4-11 muestra una visión general del Modelo de Implementación.

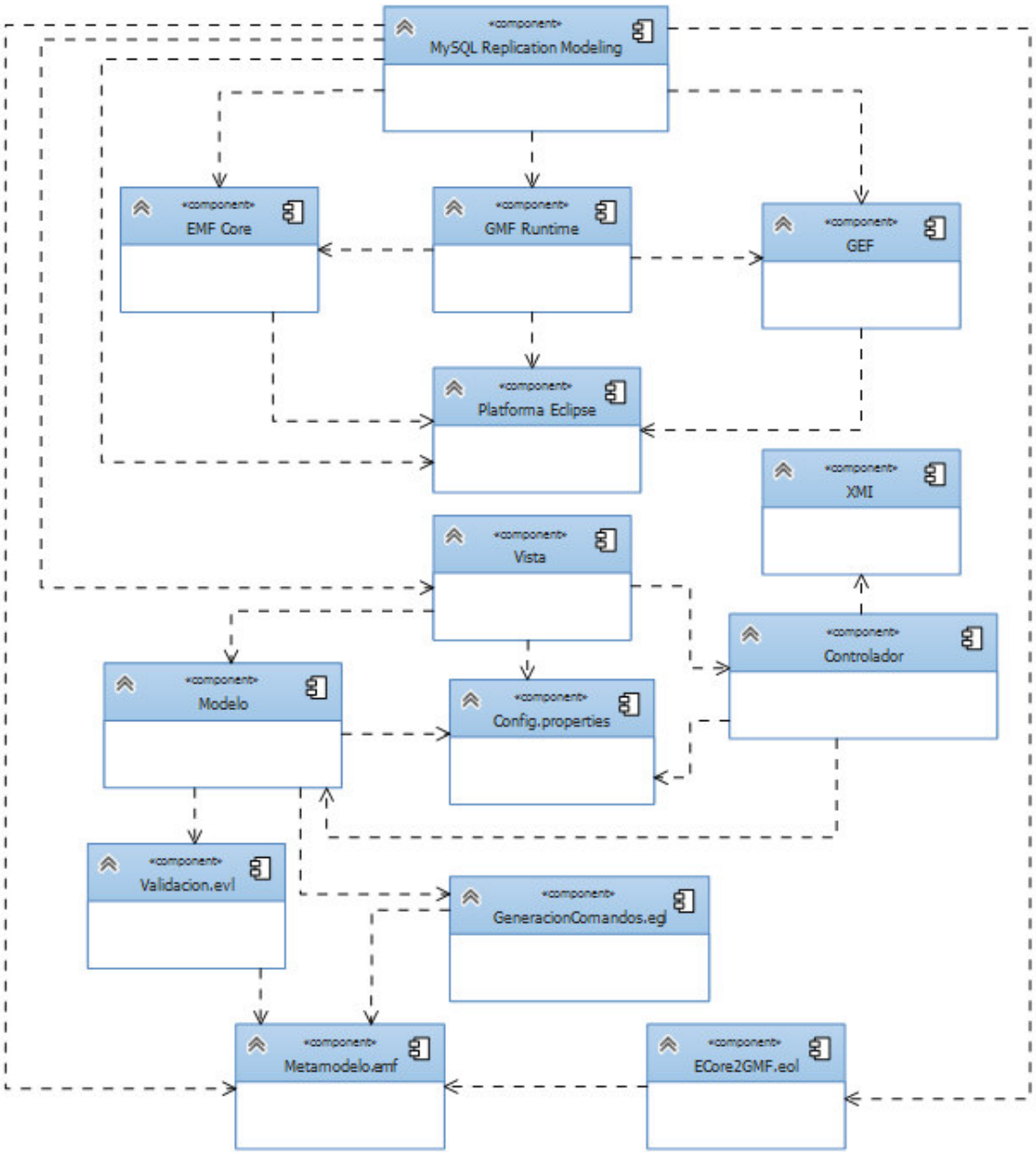


Figura 4-11 Visión General del Modelo de Componentes [Elaboración Propia]

El modelo muestra la dependencia de la herramienta MySQL Replication Modeling de la plataforma Eclipse, EMF (Eclipse Modeling Framework), GEF (Graphical Editing Framework), y el runtime de GMF (Graphical Modeling Framework).

GMF está basado en el modelo MVC (Modelo-Vista-Controlador) por lo tanto la herramienta hereda esta arquitectura. La generación del código fuente de la herramienta se realiza a partir de un metamodelo implementado con emfatic (.emf), los archivos de validación Epsilon Validation Language (.evl) se programan usando los elementos del metamodelo, así como los archivos de generación de comandos Epsilon Generation Language (.egl). Alguna de las personalizaciones estéticas del editor gráfico GMF se realiza mediante el archivo ECore2GMF.eol en la que se programa usando el lenguaje Epsilon Object Language (.eol). La persistencia de los modelos se realiza en el formato XMI (XML Metadata Interchange).

4.3.2.4 Vista de Despliegue

Esta vista muestra los nodos físicos mediante un diagrama de despliegue.

La herramienta se desplegará como una serie de plugins del IDE de eclipse, por lo tanto solo se tendrá un nodo físico puesto que la persistencia se realizará en formato XMI (XML Metadata Interchange) en el mismo nodo físico.

La Figura 4-12 muestra este único nodo físico en el diagrama de despliegue de la herramienta.

Figura 4-12 Diagrama de Despliegue de la Herramienta [Elaboración Propia]

4.3.2.5 Vista de Datos

Esta vista especifica arquitectónicamente los elementos en el Modelo de Datos.

Esta vista se refiere al almacenamiento de los procedimientos que proporcionan la persistencia de la herramienta.

La persistencia de los modelos de replicación de la herramienta se realiza en el estándar del formato XMI (XML Metadata Interchange) y no en una base de datos relacional.

A continuación se detalla la fase de construcción del proceso de desarrollo de la herramienta.

4.4 Fase de Construcción

En esta fase se detalla la arquitectura tecnológica y la disciplina de implementación de la herramienta propuesta denominada **MySQL Replication Modeling**.

4.4.1 Arquitectura Tecnológica

La arquitectura tecnológica de la herramienta está relacionado con los frameworks EMF y GMF de la plataforma Eclipse, puesto que son los frameworks seleccionados para el desarrollo de la herramienta propuesta basado en los principios de MDE. GMF está basado en el framework GEF (Graphical Editing Framework), la cual usa el patrón Modelo-Vista-Controlador (MVC), por lo tanto GMF hereda esta arquitectura. Para el desarrollo de la herramienta se usó el IDE Eclipse Kepler 4.3 para el sistema operativo Windows de 32 bits, este IDE contiene una versión estable de Epsilon (v1.1_SR1), EMF, GMF y Emfatic. La Figura 4-13 ilustra la arquitectura tecnológica.

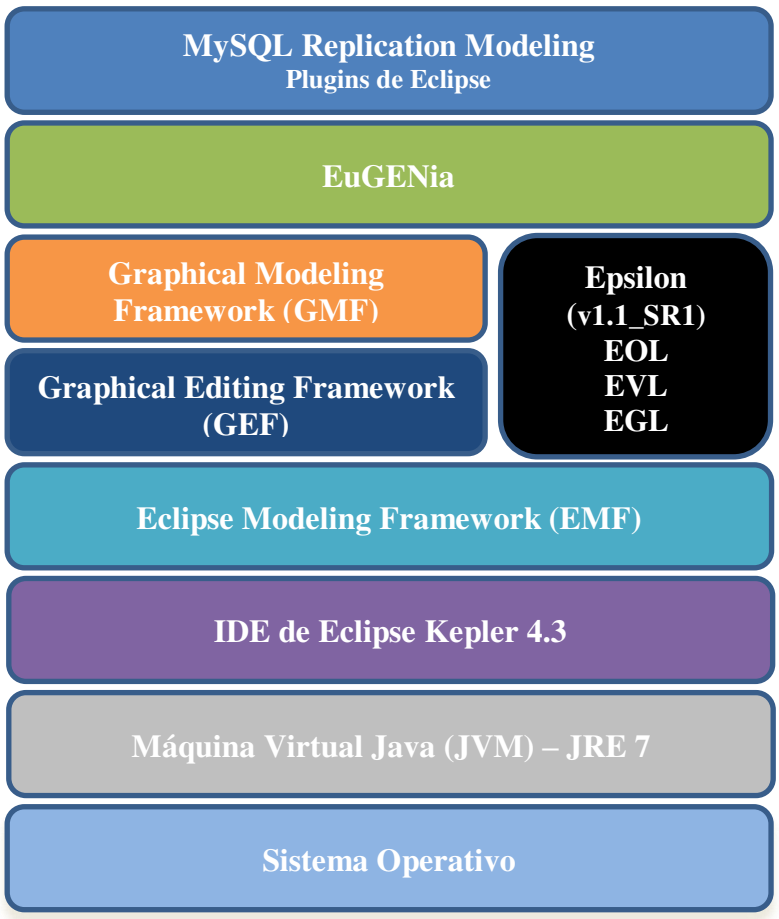


Figura 4-13 Arquitectura Tecnológica de la Herramienta [Elaboración Propia]

A continuación se describe cada uno de los componentes de la arquitectura tecnológica.

- **Sistema Operativo**: El sistema operativo donde podrá correr la herramienta propuesta. La herramienta debe ser multiplataforma, específicamente debe poder ejecutarse en el sistema operativo Windows, Linux y Mac OS.
- **Máquina Virtual de Java (JVM)**: Este componente permitirá ejecutar el código Java de la herramienta en los sistemas operativos antes mencionados.
- **Eclipse Kepler 4.3**: El IDE Eclipse en el que se desarrollará y desplegará la herramienta mediante plugins.
- **Eclipse Modeling Framework (EMF)**: Es el framework de Eclipse para definir el metamodelo (también conocido como sintaxis abstracta).
- **Graphical Modeling Framework (GMF)**: Es el framework de Eclipse para definir la representación visual del metamodelo (también conocido como sintaxis concreta). GMF está basado en el framework **Graphical Editing Framework (GEF)**, la cual usa el patrón de arquitectura Modelo-Vista-Controlador (MVC), por lo tanto GMF hereda dicho patrón.
- **Epsilon**: Es una familia de lenguajes para la administración de modelos. La herramienta propuesta hará uso de los siguientes lenguajes:
  - **Epsilon Object Language (EOL)**: Lenguaje principal de Epsilon con sintaxis parecida a Java.
  - **Epsilon Validation Language (EVL)**: Lenguaje usado para implementar las validaciones a un modelo de replicación de MySQL.
  - **Epsilon Generation Language (EGL)**: Lenguaje usado para implementar la generación de los comandos mysqlreplicate de configuración de la replicación MySQL.
- **Eugenia**: Es una herramienta desarrollada por el proyecto Epsilon de Eclipse que permite generar automáticamente un editor gráfico GMF desde un metamodelo con anotaciones en Emfatic.
- **MySQL Replication Modeling**: Es la denominación a la herramienta propuesta, que contendrá una serie de plugins con extensión .jar y serán desplegadas en el IDE Eclipse.

A continuación se detalla la disciplina de implementación.

4.4.2 Disciplina de Implementación

En esta disciplina se definieron 5 iteraciones listadas en la Tabla 4-7.

# de Iteración	Descripción
1	Implementación del Metamodelo
2	Generación automática del Editor Gráfico GMF
3	Implementación de Mejoras Estéticas al Editor Gráfico GMF
4	Implementación de la Validación de Modelos de Replicación MySQL
5	Implementación de la Generación de Comandos mysqlreplicate

Tabla 4-7 Iteraciones Realizadas en la Implementación de la Herramienta MySQL Replication Modeling [Elaboración Propia]

4.4.2.1 Primera Iteración: Implementación del Metamodelo

El primer paso es crear un proyecto GMF tal como se muestra en la Figura 4-14.

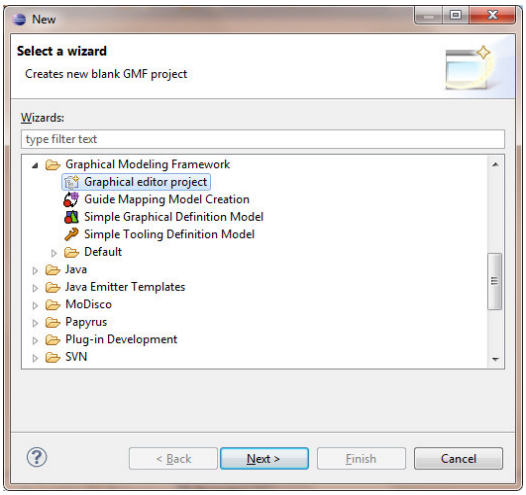


Figura 4-14 Creación de un Proyecto GMF en Eclipse [Elaboración Propia]

Luego de definir el nombre y la ruta del proyecto GMF, aparecerá un nuevo proyecto GMF vacío en la que se va a empezar a trabajar la codificación. La pantalla del proyecto GMF se puede visualizar en la Figura 4-15.

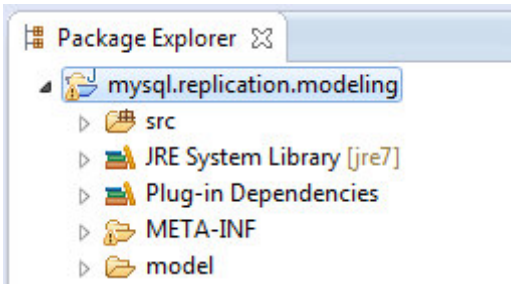


Figura 4-15 Proyecto GMF Vacío [Elaboración Propia]

En este proyecto GMF se crea por defecto una carpeta llamada model.

Dentro de la carpeta model se va a agregar el archivo Emfatic para definir el metamodelo. La Figura 4-16 es la pantalla para agregar un Emfatic.

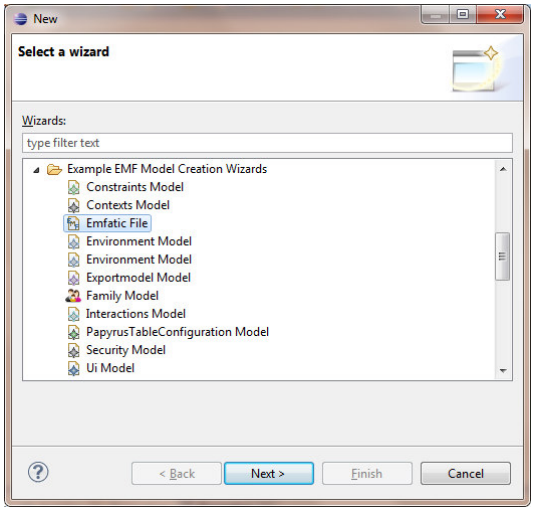


Figura 4-16 Creación de Archivo Emfatic [Elaboración Propia]

Seguidamente aparecerá el archivo Emfatic, tal como se muestra en la Figura 4-17.

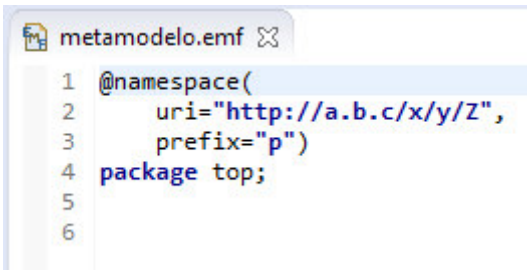


Figura 4-17 Archivo Emfatic Creado [Elaboración Propia]

En el archivo Emfatic se va a iniciar la codificación del metamodelo y las anotaciones que será luego leído por la herramienta Eugenia para generar automáticamente el editor gráfico GMF. La Figura 4-18 muestra el código del metamodelo anotado.

metamodel.emf

```
@namespace(uri="http://www.unmsm.edu.pe/mysql_replication_modeling",
prefix="mysql_replication_modeling")
package mysql_replication_modeling;

@gmf.diagram(model.extension="rplm", diagram.extension="rpl",
onefile="true")
class Model {
  val MySQLServer[*] mysql_servers;
  val MasterSlaveRelationship[*] relationships_ms;
  val MasterMasterRelationship[*] relationships_mm;
}

@gmf.node(figure="mysql.replication.modeling.figures.MySQLServerFigure",
label="host, port", label.pattern="{0} : {1}", label.icon="false",
label.placement="external")
class MySQLServer
{
  attr String host = "server";
  attr String port = "3306";
}

@gmf.link(source="master", target="slave", style="solid", label.icon="true",
width="1", color="0,0,255", target.decoration="arrow",
source.constraint="self <> oppositeEnd")
class MasterSlaveRelationship {
  ref MySQLServer master;
  ref MySQLServer slave;
}

@gmf.link(source="masterSource", target="masterTarget", style="solid",
label.icon="true", width="1", color="255,0,0", source.decoration="arrow",
target.decoration="arrow", source.constraint="self <> oppositeEnd")
class MasterMasterRelationship {
  ref MySQLServer masterSource;
  ref MySQLServer masterTarget;
}
```

Figura 4-18 Archivo Emfatic Anotado del Metamodelo [Elaboración Propia]

En el metamodelo se ha definido una serie de anotaciones con la que le decimos a la herramienta Eugenia cómo mostrar los elementos del metamodelo.

A continuación se describe los elementos del código del metamodelo.

- **package**: El metamodelo (EPackage en la terminología EMF) es definido con el nombre "mysql_replication_modeling", la cual contiene todos los elementos (nodos y enlaces) definidos en el metamodelo. Un nodo es una figura que puede ser o no visible en un diagrama GMF, y los enlaces son las relaciones entre los nodos. Por ejemplo, el nodo MySQLServer representa a un servidor MySQL y los enlaces MasterSlaveRelationship y MasterMasterRelationship representan las relaciones maestro-esclavo y maestro-maestro respectivamente.

- **class:** La palabra clave “class”, así como en Java, es usado en Emfatic para declarar una meta-clase (EClass en la terminología EMF). Los atributos en Emfatic son definidos usando la palabra clave “attr” seguido por el tipo de dato y el nombre del atributo.
- **@gmf.diagram**: La metaclase **Model** representa el elemento raíz, que es el diagrama que contendrá todo el modelo con los nodos y enlaces entre nodos. Usando la anotación *@gmf.diagram()* denotamos la metaclase “Model” como el elemento raíz. Los nodos y enlaces del elemento raíz del metamodelo son referencias containment especificadas usando la palabra clave “val”, esto quiere decir que si el diagrama es eliminado, los nodos y enlaces también lo serán.
- **@gmf.node**: Con esta anotación se define una metaclase como nodo. Los parámetros en esta anotación especifican la etiqueta que mostrará el nodo.
- **@gmf.link**: Con esta anotación se define una metaclase como enlace. Los parámetros en esta notación especifican el nodo origen y nodo destino del enlace, así como el color, tipo de decoración del enlace, entre otras características. Cada una de las meta-clases de tipo enlace referencian a la meta-clase MySQLServer de tipo nodo para especificar su origen y destino, estas son referencias normales y se definen con la palabra clave “ref”.

4.4.2.2 Segunda Iteración: Generación automática del Editor Gráfico GMF

En esta iteración se va a mostrar en detalle los pasos para generar el editor gráfico GMF con la herramienta Eugenia.

Para generar el editor GMF, Eugenia necesita como entrada el metamodelo Emfatic que se desarrolló en la primera iteración.

Eugenia se encargará de generar el Modelo de Herramientas (Tooling Model), el Modelo de Definición Gráfica (Graph Definition Model) y el Modelo de Mapeo (Mapping Model) requeridos por el framework GMF.

Luego Eugenia realiza la unión (merge) de estos tres modelos para generar así el Modelo de Generación (Gen Model) para finalmente generar el código fuente del editor gráfico GMF como un proyecto plugin de Eclipse.

En la Figura 4-19 se ilustra el proceso de generación de un editor gráfico GMF con la herramienta Eugenia.

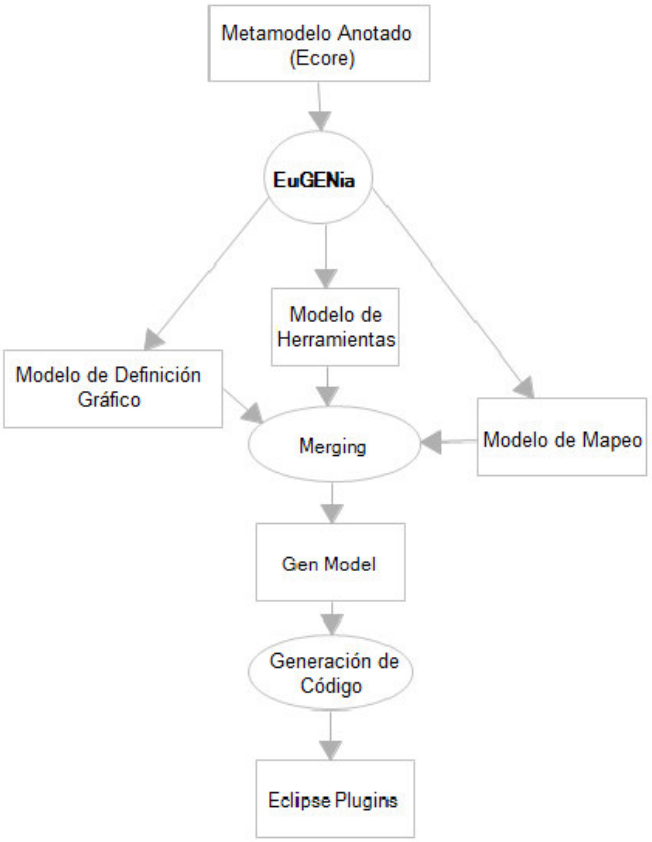


Figura 4-19 Resumen del Proceso de Generación Automática de un Editor GMF con Eugenia [Kolovos+ 2009a]

Entonces, se va proceder a generar con Eugenia desde el archivo .emf (Emfatic) del metamodelo la primera versión del editor gráfico GMF. La Figura 4-20 ilustra la pantalla para generar el editor gráfico GMF con Eugenia.

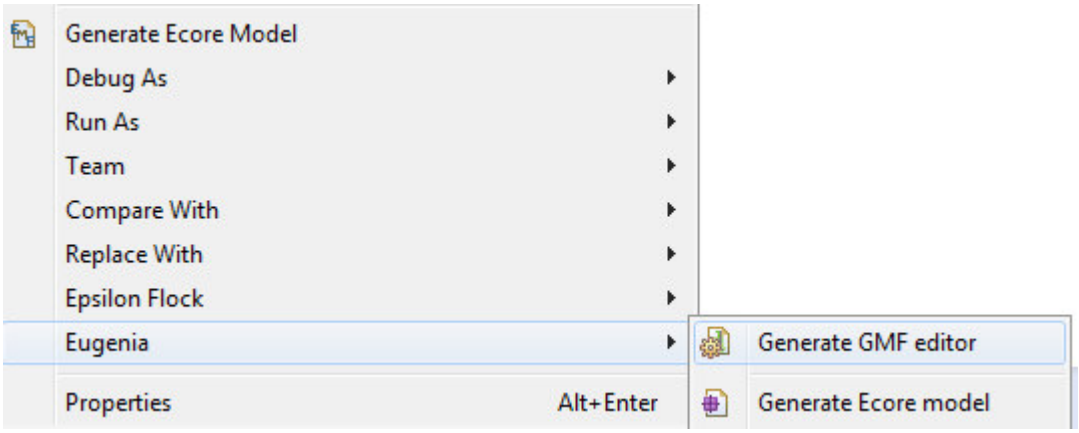


Figura 4-20 Generación del Editor Gráfico GMF con la Herramienta Eugenia [Elaboración Propia]

Luego aparecerá una barra indicando el progreso de la generación del editor gráfico, esta imagen se muestra en la Figura 4-21.

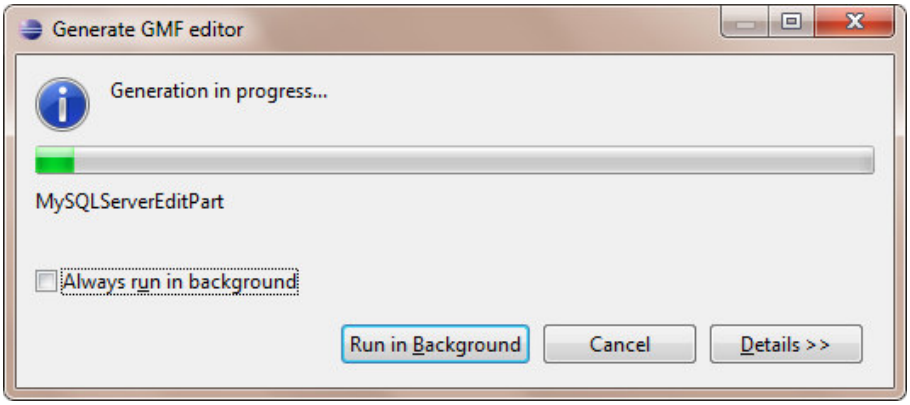


Figura 4-21 Barra de Progreso de la Generación del Editor Gráfico con Eugenia [Elaboración Propia]

Al finalizar la generación, Eugenia muestra en pantalla el mensaje de confirmación mostrado en la Figura 4-22.

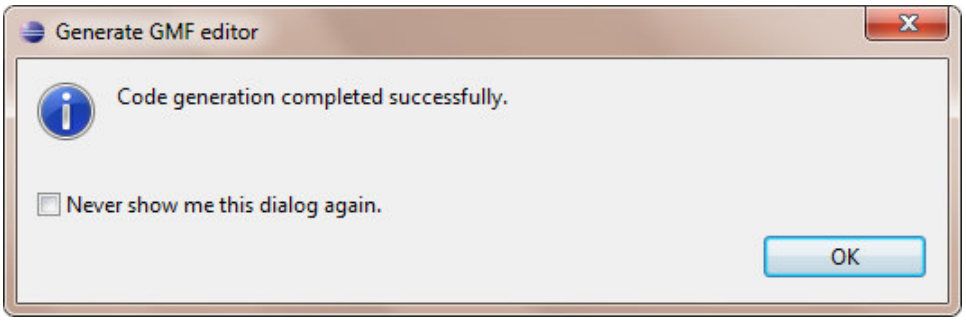


Figura 4-22 Confirmación de Eugenia de la Generación Completa del Editor GMF [Elaboración Propia]

En este punto, Eugenia ha generado dentro del proyecto GMF que contiene el metamodelo los archivos .gmfgraph (Modelo de Definición Gráfica), .gmftool (Modelo de Herramientas), .gmfmap (Modelo de Mapeo), .genmodel (Modelo de Generación) y .gmfgen. En base a estos archivos se ha generado el código fuente del editor gráfico GMF.

En el proyecto GMF que contiene el metamodelo se crearon tres paquetes con código fuente. También se crearon otros proyectos de tipo plugin que contienen el resto del código fuente del editor gráfico GMF. Los plugins creados son el .diagram, .edit, .editor y .tests. La Figura 4-23 muestra los archivos y proyectos generados automáticamente por Eugenia.

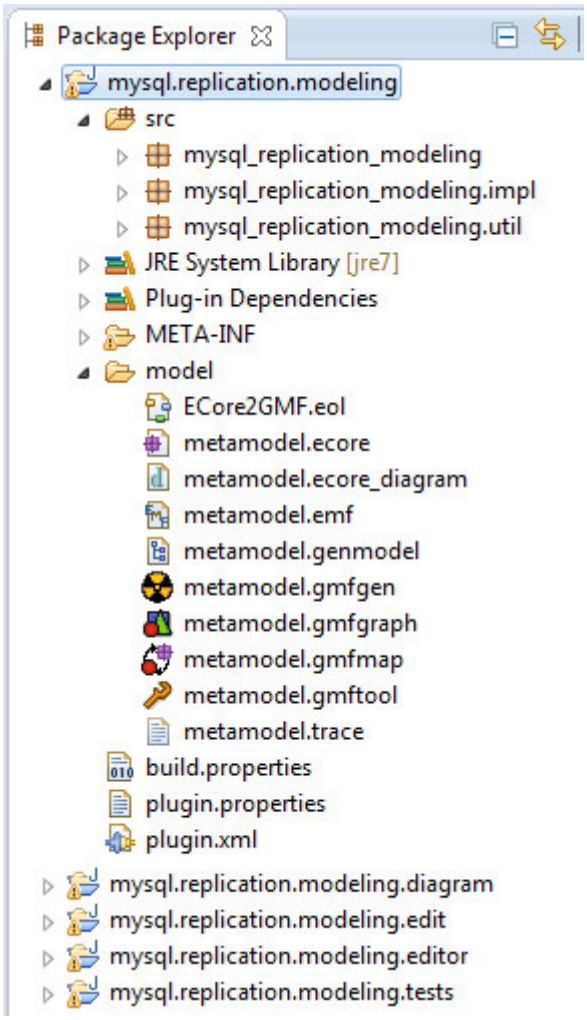


Figura 4-23 Plugins Generados por la Herramienta Eugenia [Elaboración Propia]

La Figura 4-24 muestra la pantalla del archivo .gmfgraph (Modelo de Definición Gráfica).

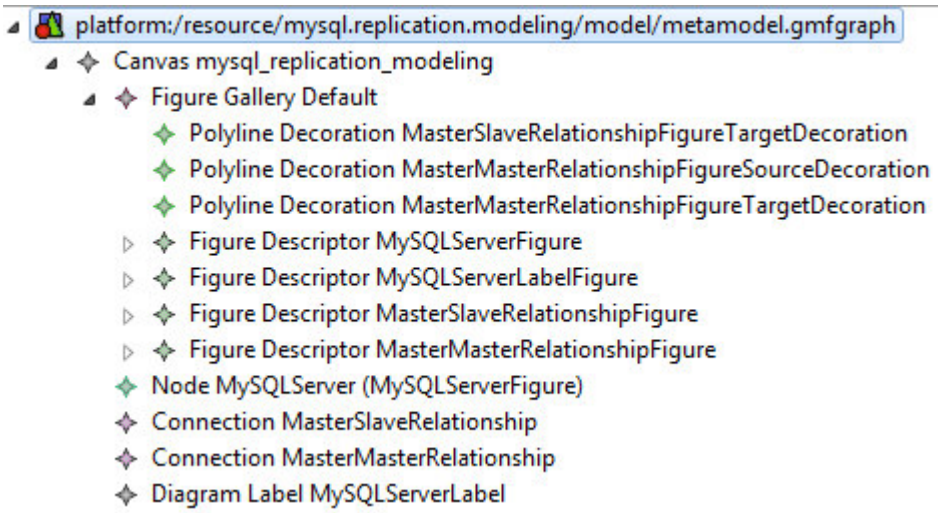


Figura 4-24 Archivo .gmfgraph Generado [Elaboración Propia]

La Figura 4-25 muestra el archivo .gmftool (Modelo de Herramientas).

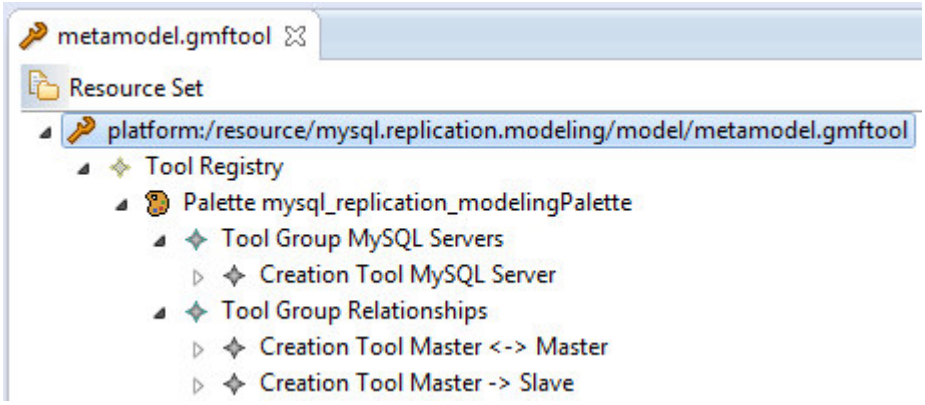


Figura 4-25 Archivo .gmftool Generado [Elaboración Propia]

La Figura 4-26 muestra la pantalla del archivo .gmfmap (Modelo de Mapeo).

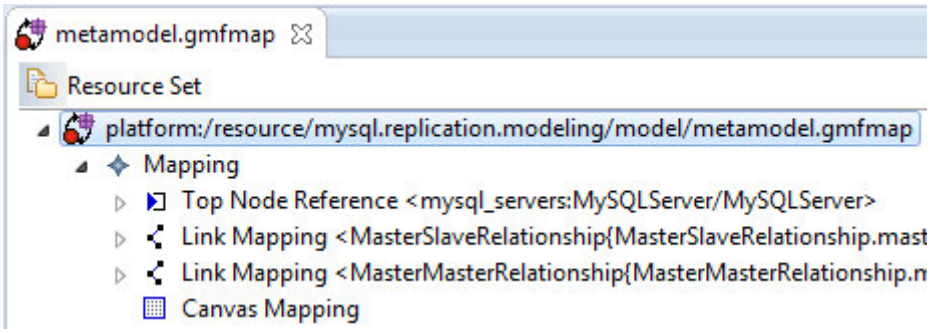


Figura 4-26 Archivo .gmfmap Generado [Elaboración Propia]

El proyecto generado .diagram es el plugin que contiene la lógica principal del editor gráfico GMF. La pantalla del proyecto .diagram se muestra en la Figura 4-27.

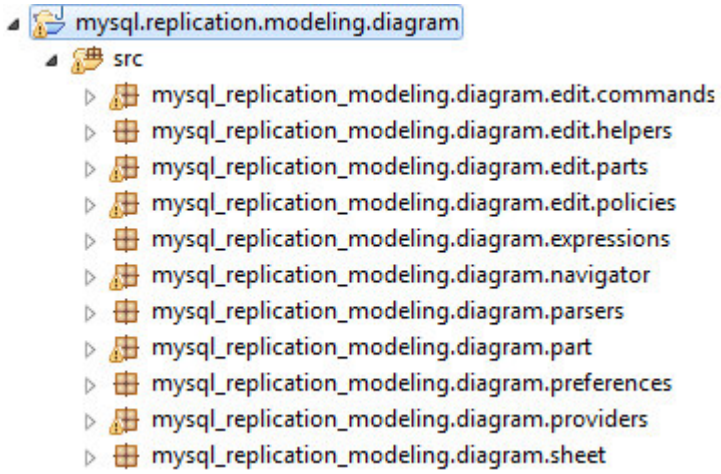


Figura 4-27 Plugin .diagram del Editor Gráfico GMF [Elaboración Propia]

El plugin .diagram tiene varios paquetes la cual se va a describir brevemente a continuación los principales.

EditParts son los elementos principales en el editor. Las clases en el paquete "mysql_replication_modeling.diagram.edit.parts" son responsables del mapeo entre los elementos del modelo y las figuras visuales, así como en el comportamiento de las figuras. Dos tipos de editparts son usados. "GraphicalEditParts" están siendo extendidos por el diagrama y los nodos y "ConnectionEditParts" están siendo extendidos por los enlaces o links.

Los EditPolicies son los responsables de manejar los requests y crear un comando para cada request a ser ejecutado. Los EditPolices son los encargados de llevar la funcionalidad de edición a los EditParts. Los comandos son respuestas a los requests. Otro paquete importante es el "mysql_replication_modeling.diagram.part", este paquete provee clases que soportan la manipulación de los diagramas, tales como abrir, crear y guardar un diagrama.

La pantalla de la herramienta en esta segunda iteración se muestra en la Figura 4-28.

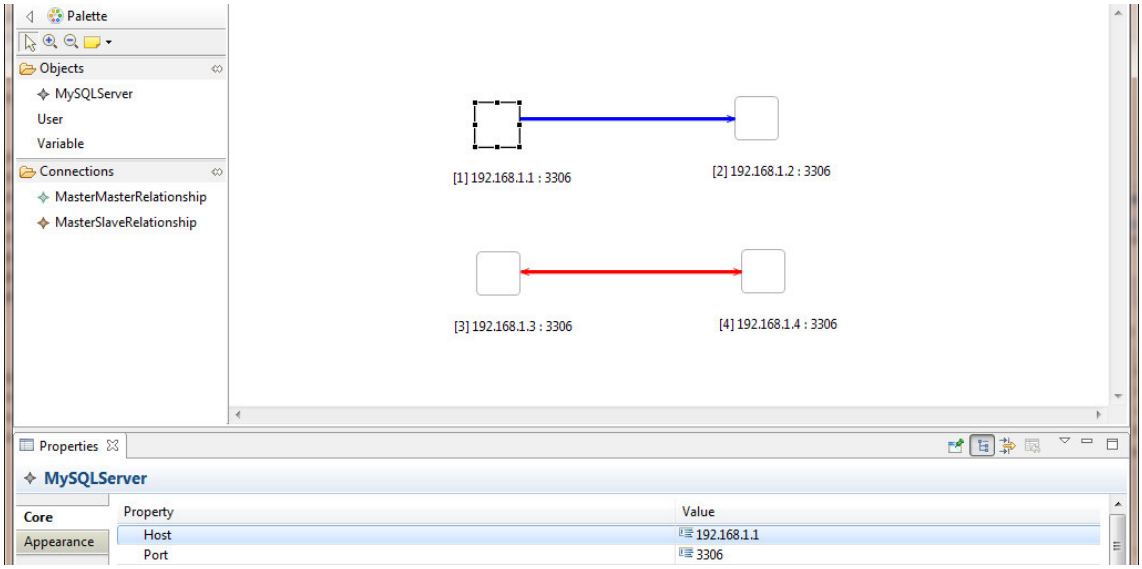


Figura 4-28 Editor Gráfico Generado por Eugenia [Elaboración Propia]

La funcionalidad y estética de esta primera versión de la herramienta es muy limitada. Se requiere desarrollar personalizaciones, para esto es necesario cambiar el código fuente generado, así como extender la funcionalidad con el desarrollo e integración de nuevos plugins conocidos en Eclipse como puntos de extensión, esta tarea no es trivial y consume tiempo. Estas personalizaciones se detallarán en las siguientes iteraciones.

4.4.2.3 Tercera Iteración: Implementación de Mejoras Estéticas al Editor Gráfico GMF

En esta tercera iteración se va a personalizar el editor gráfico GMF generado por Eugenia.

Las anotaciones disponibles son limitadas y no permiten obtener un editor gráfico para características complejas, es por ello, Eugenia también permite a los desarrolladores implementar tres modelos textuales de personalización (ECore2GMF.eol, FixGenModel.eol y FixGMFGen.eol) descritos en el lenguaje EOL (Epsilon Object Language).

Para personalizar esta herramienta solo se consideró necesario agregar y codificar el archivo ECore2GMF.eol.

Este archivo se ejecuta cada vez que se generan los modelos .gmfgraph, .gmftool y .gmfmap, por lo que se emplea para la personalización de dichos modelos. Además, ofrece la posibilidad de interactuar con el modelo de dominio Ecore.

La Figura 4-29 muestra la pantalla para agregar un archivo EOL, el cual se debe nombrar ECore2GMF.eol.

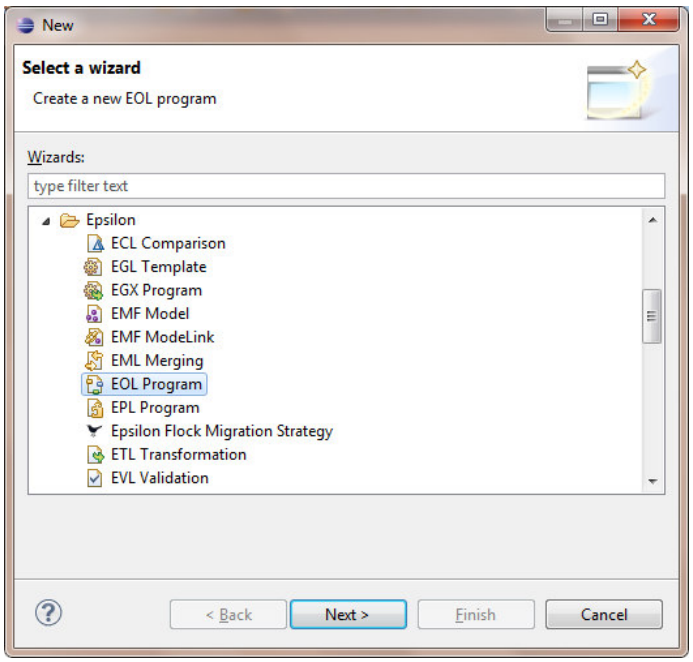


Figura 4-29 Agregar Archivo Ecore2GMF.eol [Elaboración Propia]

La Figura 4-30 muestra el código EOL desarrollado para personalizar la paleta de herramientas del editor gráfico GMF.

ECore2GMF.eol

```
---------
-- Tool
---------
var palette = GmfTool!Palette.all.selectOne(n|n.title='mysql_replication_modelingPalette');
var objectsToolGroup = GmfTool!ToolGroup.all.selectOne(t|t.title = 'Objects');
var connectionsToolGroup = GmfTool!ToolGroup.all.selectOne(t|t.title = 'Connections');

-- Create ServersGroup
var serversGroup = new GmfTool!ToolGroup;
serversGroup.title='MySQL Servers';
serversGroup.collapsible=true;
palette.tools.add(serversGroup);
var mySQLServerTool = objectsToolGroup.tools.selectOne(n|n.title = 'MySQLServer');
mySQLServerTool.title='MySQL Server';
mySQLServerTool.description='Create a new MySQL Server Instance';
serversGroup.tools.add(mySQLServerTool);

--Create LinksGroup
var linksGroup=new GmfTool!ToolGroup;
linksGroup.title='Relationships';
linksGroup.collapsible=true;
palette.tools.add(linksGroup);
var multiMasterLinkTool = connectionsToolGroup.tools.selectOne(n|n.title = 'MasterMasterRelationship');
multiMasterLinkTool.title='Master <-> Master';
multiMasterLinkTool.description='Create a Multi-Master Relationship';
linksGroup.tools.add(multiMasterLinkTool);
var masterSlaveLinkTool = connectionsToolGroup.tools.selectOne(n|n.title = 'MasterSlaveRelationship');
masterSlaveLinkTool.title='Master -> Slave';
masterSlaveLinkTool.description='Create a Master-Slave Relationship';
linksGroup.tools.add(masterSlaveLinkTool);


--Delete ObjectGroup
palette.tools.remove(objectsToolGroup);
--Delete ConnectionGroup
palette.tools.remove(connectionsToolGroup);
```

Figura 4-30 Archivo Ecore2GMF.eol [Elaboración Propia]

Al volver a generar el editor GMF con Eugenia, se obtendrá la paleta de herramientas mostrada en la Figura 4-31.

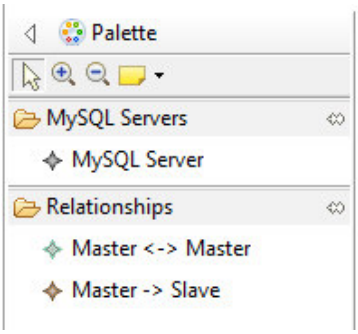


Figura 4-31 Paleta de Herramientas personalizada con el archivo ECore2GMF.eol [Elaboración Propia]

Otro punto importante es que los nodos que representan las figuras de los servidores MySQL se muestren con figuras representativas. Para lograr esto, primero tenemos que generar un plugin desde el archivo .gmfgraph tal como se puede observar en la Figura 4-32.

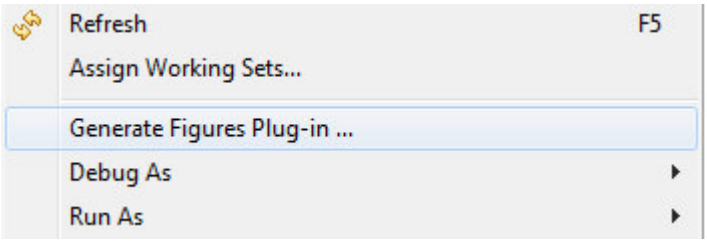


Figura 4-32 Generación de Plugin de Figuras [Elaboración Propia]

Seguidamente, se generará el plugin mostrado en la Figura 4-33.

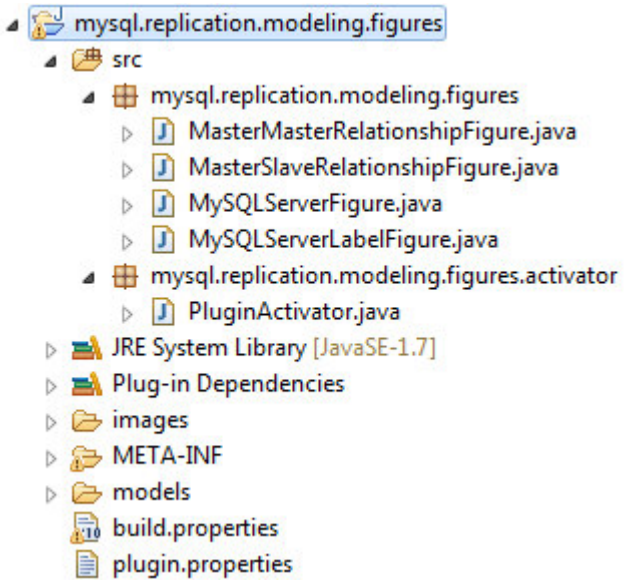


Figura 4-33 Plugin de Figuras Generado [Elaboración Propia]

Al plugin de figuras generado, se le debe agregar las dependencias mostradas en la Figura 4-34.

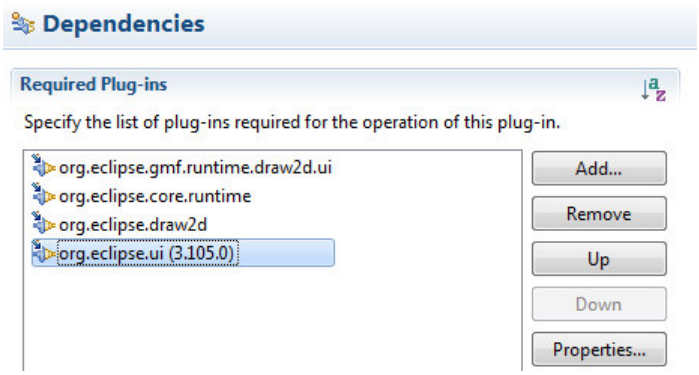


Figura 4-34 Dependencias del Plugin de Figuras [Elaboración Propia]

Se tiene que modificar la clase PluginActivator para que herede de la clase AbstractUIPlugin, tal como se puede ver en la Figura 4-35.

PluginActivator.java

```
1  package mysql.replication.modeling.figures.activator;
2
3  import org.eclipse.ui.plugin.AbstractUIPlugin;
5
6  /**
7   * @generated
8   */
9  public class PluginActivator extends AbstractUIPlugin
```

Figura 4-35 Clase PluginActivator del Plugin de Figuras [Elaboración Propia]

Como siguiente paso se modificará la clase MySQLServerFigure para indicar la imagen a mostrar, tal como se muestra en la Figura 4-36.

MySQLServerFigure.java

```
1  package mysql.replication.modeling.figures;
2
3  import mysql.replication.modeling.figures.activator.PluginActivator;
5
6  /**
7   * @generated
8   */
9  public class MySQLServerFigure extends ImageFigure {
10
11     /**
12      * @generated
13      */
14     public MySQLServerFigure() {
15         super(PluginActivator.imageDescriptorFromPlugin(
16                 PluginActivator.ID,
17                 "images/MySQLServer.png").createImage(), 0);
18     }
19 }
```

Figura 4-36 Código de la Clase MySQLServerFigure del Plugin de Figuras [Elaboración Propia]

Luego de los cambios realizados en la clase MySQLServerFigure, se tiene que modificar el archivo emfatic del metamodelo para indicarle a Eugenia la figura que se mostrará en el nodo MySQLServer, estos cambios son mostrados en la Figura 4-37.

```
13 @gmf.node(figure="mysql.replication.modeling.figures.MySQLServerFigure",
14  label="host, port", label.pattern="{0} : {1}", label.icon="false",
15  label.placement="external")
16  class MySQLServer
17  {
18    attr String host = "server";
19    attr String port = "3306";
20  }
```

Figura 4-37 Modificación del Archivo Emfatic [Elaboración Propia]

En este punto se debe generar nuevamente el editor con Eugenia. Luego de esta nueva generación del editor, se tiene que modificar el plugin .diagram para agregar la dependencia al plugin .figures. En la Figura 4-38 se puede ver la pantalla para agregar esta dependencia.

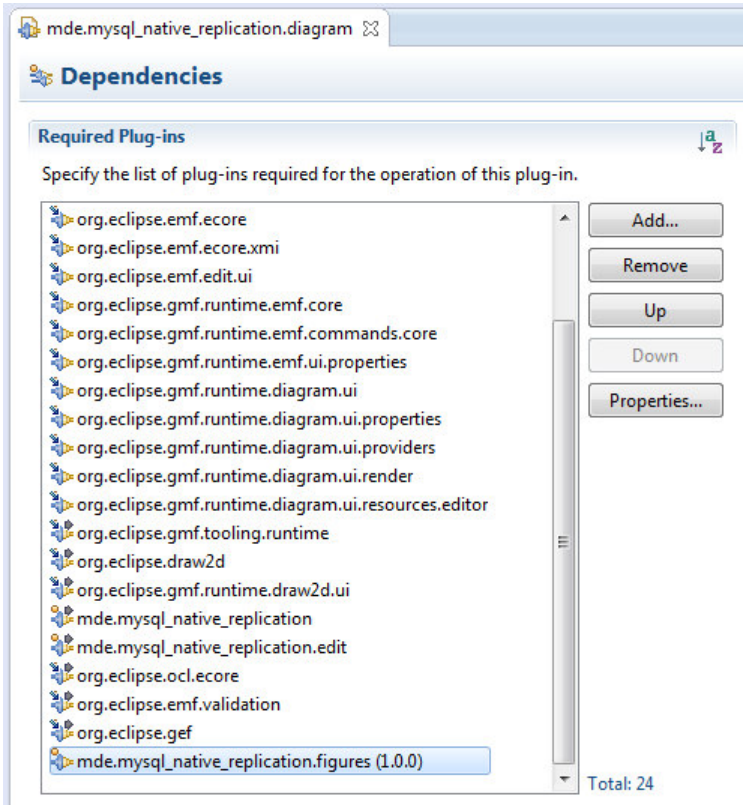


Figura 4-38 Agregar Dependencia del plugin de Figuras al Plugin Diagram [Elaboración Propia]

Para cambiar los iconos de la paleta de herramientas, se tiene que reemplazar los archivos .gif creados en el plugin .edit, tal como se muestra en la Figura 4-39.

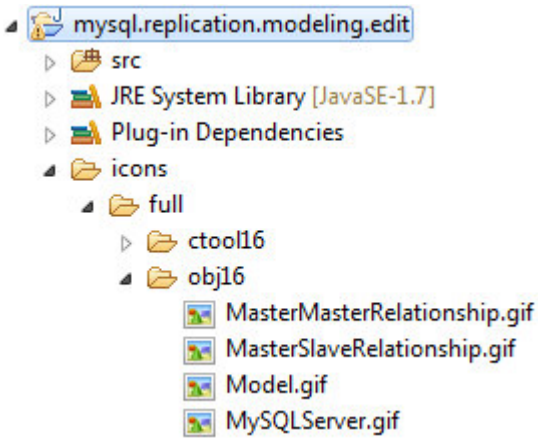


Figura 4-39 Iconos de la Paleta de Herramientas [Elaboración Propia]

Otro cambio importante es modificar las etiquetas de los archivos .properties de los plugins generados. Un ejemplo del archivo messages.properties del plugin .diagram se muestra en la Figura 4-40.

messages.properties

```
# TODO: manually put keys and values
Mysql_replication_modelingCreationWizardTitle=New MySQL Replication Model
Mysql_replication_modelingCreationWizard_DiagramModelFilePageTitle=Create MySQL Replication Model
Mysql_replication_modelingCreationWizard_DiagramModelFilePageDescription=Select file that will contain diagram model.
Mysql_replication_modelingCreationWizard_DomainModelFilePageTitle=Create MySQL Replication Domain Model
Mysql_replication_modelingCreationWizard_DomainModelFilePageDescription=Select file that will contain domain model.
Mysql_replication_modelingCreationWizardOpenEditorError=Error opening diagram editor
Mysql_replication_modelingCreationWizardCreationError=Creation Problems
Mysql_replication_modelingCreationWizardPageExtensionError=File name should have {0} extension.
Mysql_replication_modelingDiagramEditorUtil_OpenModelResourceErrorDialogTitle=Error
Mysql_replication_modelingDiagramEditorUtil_OpenModelResourceErrorDialogMessage=Failed to load model file {0}
Mysql_replication_modelingDiagramEditorUtil_CreateDiagramProgressTask=Creating diagram and model files
Mysql_replication_modelingDiagramEditorUtil_CreateDiagramCommandLabel=Creating diagram and model
Mysql_replication_modelingDocumentProvider_isModifiable=Updating cache failed
Mysql_replication_modelingDocumentProvider_handleElementContentChanged=Failed to refresh hierarchy for changed resource
Mysql_replication_modelingDocumentProvider_IncorrectInputError={1}
Mysql_replication_modelingDocumentProvider_NoDiagramInResourceError=Diagram is not present in resource
Mysql_replication_modelingDocumentProvider_DiagramLoadingError=Error loading diagram
Mysql_replication_modelingDocumentProvider_UnsynchronizedFileSaveError=The file has been changed on the file system
Mysql_replication_modelingDocumentProvider_SaveDiagramTask=Saving diagram
```

Figura 4-40 Etiquetas del Archivo messages.properties del Plugin .diagram [Elaboración Propia]

Con el fin de poder invocar las opciones de la herramienta, se tuvo que crear un plugin llamado custom para agregar un menú con estas opciones. La Figura 4-41 muestra la pantalla para crear el plugin custom.

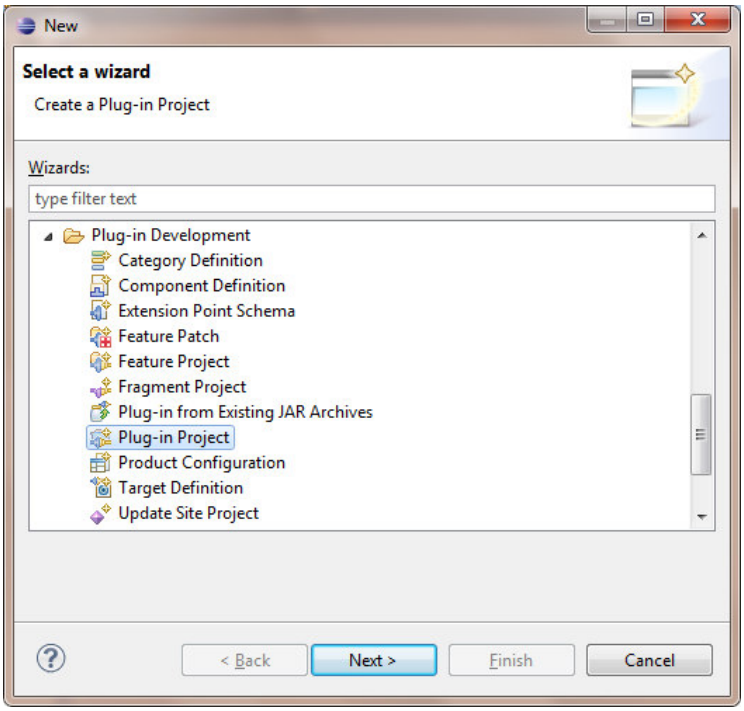


Figura 4-41 Crear Plugin .custom para Personalizaciones [Elaboración Propia]

La Figura 4-42 muestra la configuración del menú en el plugin custom.

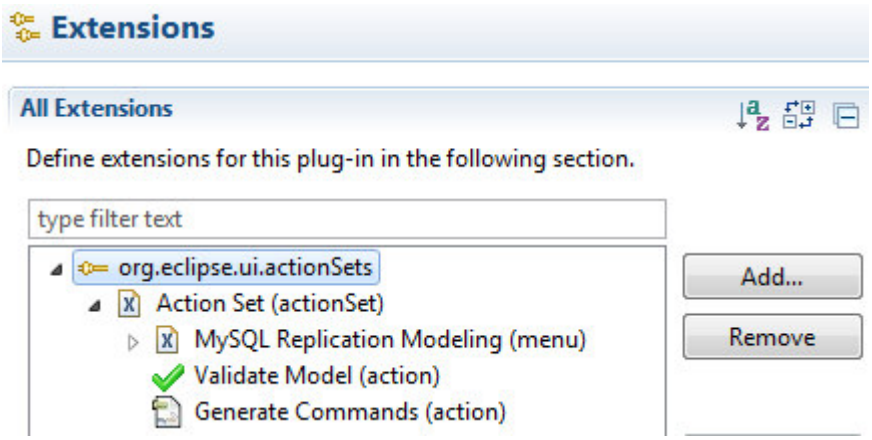


Figura 4-42 Menú Personalizado de la Herramienta [Elaboración Propia]

La Figura 4-43 muestra la interfaz de la herramienta con las personalizaciones realizadas hasta el momento, y en la que se puede notar el menú agregado en el paso anterior.

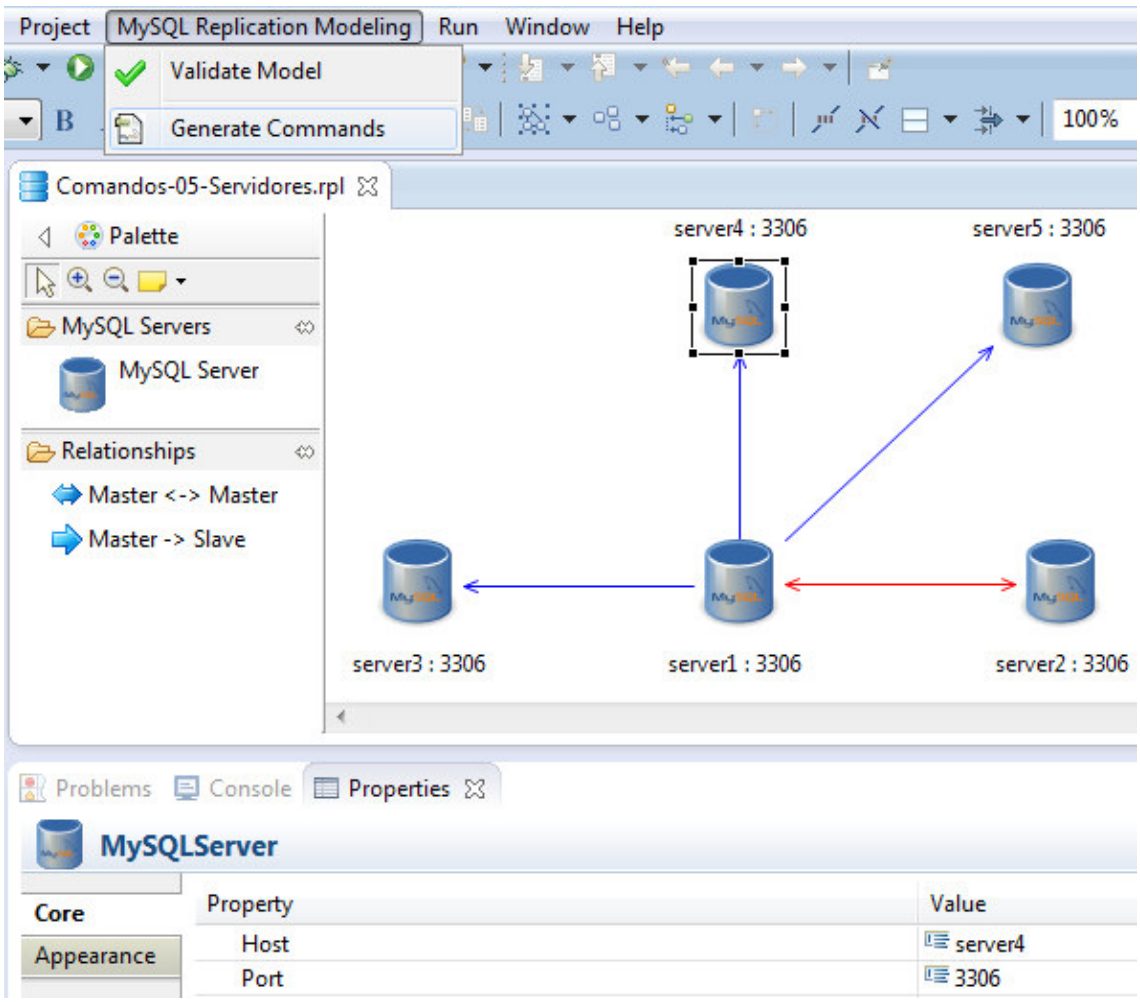


Figura 4-43 Interfaz de la Herramienta Terminada la Tercera Iteración [Elaboración Propia]

4.4.2.4 Cuarta Iteración: Implementación de la Validación de Modelos de Replicación MySQL

En esta cuarta iteración se va a desarrollar validaciones al metamodelo, de modo tal de asegurar tener modelos válidos de la replicación de MySQL. Para el desarrollo de estas validaciones se hará uso del lenguaje Epsilon Validation Language (EVL), que está basado en el lenguaje Epsilon Object Language (EOL) y permite definir restricciones de forma similar al lenguaje Object Constraint Language (OCL), pero además soporta dependencias entre las restricciones (si no se cumple la restricción A, no se evalúa la restricción B), personalizar los mensajes de error que recibe el usuario y definir (usando EOL) soluciones rápidas a los problemas de validación. Además, incorpora otra ventaja respecto de OCL, que es la posibilidad de establecer restricciones entre modelos.

Se debe asegurar tener activadas las propiedades Validation Decorators y Validation Enabled del archivo .gmfgen, de lo contrario las validaciones no se ejecutarán, estas propiedades se muestran en la Figura 4-44.

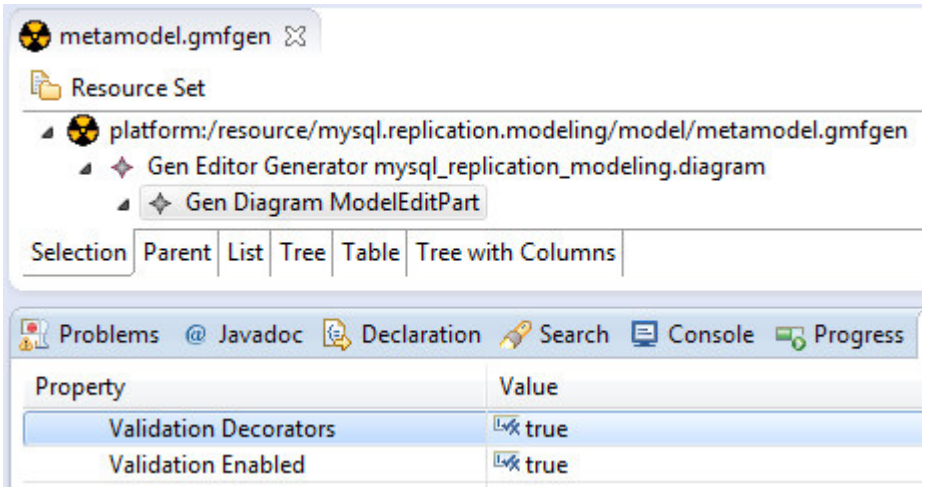


Figura 4-44 Habilitar las Validaciones en el Archivo .gmfgen [Elaboración Propia]

Luego se crea el plugin de validaciones tal como se muestra en la Figura 4-45.

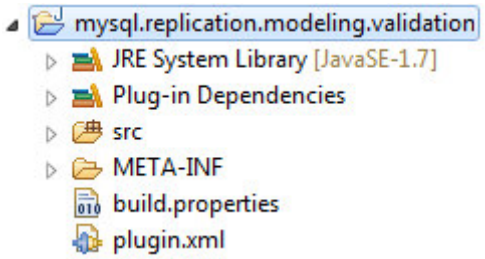


Figura 4-45 Plugin para la Validación de un Modelo [Elaboración Propia]

En este plugin de validaciones, se debe agregar la dependencia del plugin "org.eclipse.epsilon.evl.emf.validation" tal como se muestra en la Figura 4-46.

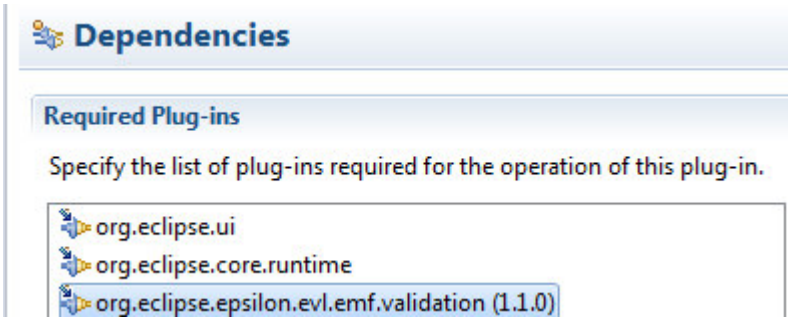


Figura 4-46 Agregar Dependencia de Epsilon EVL al Plugin de Validaciones [Elaboración Propia]

También se tiene que exportar el paquete Java del plugin para que pueda ser utilizado por los demás plugins de la herramienta, esta pantalla se muestra en la Figura 4-47.

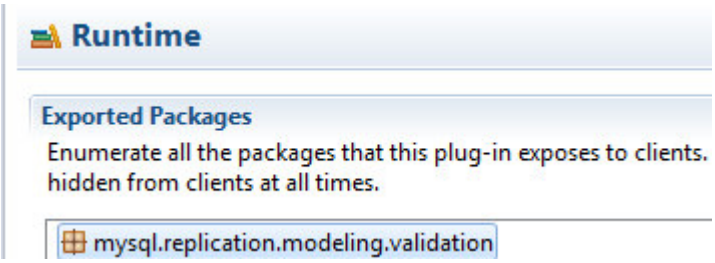


Figura 4-47 Exportar Paquete de Validación [Elaboración Propia]

Ahora se procede a agregar los archivos EVL. La Figura 4-48 muestra la pantalla para agregar un archivo EVL.

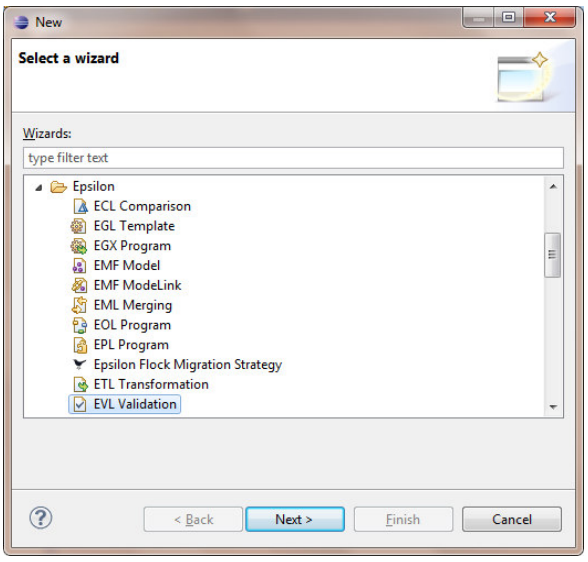


Figura 4-48 Agregar un Archivo EVL [Elaboración Propia]

El código de los archivos .evl ha sido desarrollado en base a las restricciones establecidas en la documentación oficial de MySQL.

La Figura 4-49 muestra el código del archivo Validations.evl desarrollado para el contexto de la metaclase MySQLServer.

Validations.evl

```
import 'Operations.evl';
context MySQLServer
{
    constraint HostNotEmpty{
        check : self.host.isDefined() and self.host.trim().length > 0
        message : 'This server (' + self.host + ' : ' + self.port + ') must have a host'
    }
    constraint ValidPort{
        check : self.port.isDefined() and self.port.trim().length > 0 and isValidPort(self.port.trim())
        message : 'This server (' + self.host + ' : ' + self.port + ') must have a port number between 0 and 65535'
    }
    constraint UniqueHostAndPort{
        guard : self.satisfies('ValidPort') and self.satisfies('HostNotEmpty')
        check: MySQLServer.allInstances.forAll(i|i.host = self.host and i.port = self.port implies i=self)
        message: 'This server (' + self.host + ' : ' + self.port + ') must have unique the union between host and port'
    }
    constraint MustHaveOneAndOnlyOneMaster{
        check {
            var masters = MasterSlaveRelationship.allInstances.select(i|i.slave = self and i.master <> i.slave).size();
            if(masters < 2)
               masters = masters +
               MasterMasterRelationship.allInstances.select(i|(i.masterSource = self or i.masterTarget = self) and
                                                              i.masterSource <> i.masterTarget).size();
            return (masters <= 1);
        }
        message: 'This server (' + self.host + ' : ' + self.port + ') must have one and only one master'
    }
    constraint HasARelationship{
        check:
            MasterSlaveRelationship.allInstances.exists(r1|(r1.master = self or r1.slave = self) and r1.master <> r1.slave)
            or
            MasterMasterRelationship.allInstances.exists(r2|(r2.masterSource = self or r2.masterTarget = self) and
                                                           r2.masterSource <> r2.masterTarget)
        message: 'This server (' + self.host + ' : ' + self.port +
                 ') must have one replication relationship with other mysql server database'
    }
}
```

Figura 4-49 Código EVL para el Contexto de la Metaclase MySQLServer [Elaboración Propia]

El archivo Validations.evl mostrado en la Figura 4-49 invoca a otro archivo llamado Operations.evl la cual contiene una operación desarrollada para validar el número de puerto de un servidor MySQL, la cual es mostrado en la Figura 4-50.

Operations.evl

```
operation isValidPort(port:String) : Boolean
{
    return port.isInteger() and port.asInteger() >= 0 and port.asInteger() <= 65535;
}
```

Figura 4-50 Archivo EVL para validar el puerto del servidor [Elaboración Propia]

La Figura 4-51 muestra el código EVL para el contexto de la metaclase MasterSlaveRelationship.

Validations.evl

```
context MasterSlaveRelationship
{
    constraint MustHaveOnlyOneMaster
    {
        check {
            var masters =
                MasterSlaveRelationship.allInstances.select(i|i.slave = self.slave and
                                                            i.master <> i.slave).size();
            if(masters < 2)
               masters = masters +
                MasterMasterRelationship.allInstances.select(i|(i.masterSource = self.slave or
                                                                i.masterTarget = self.slave) and
                                                                i.masterSource <> i.masterTarget).size();
            return (masters <= 1);
        }
        message: 'The slave (' + self.slave.host + ' : ' + self.slave.port + ') must have one and only one master'
    }

    constraint ExistsOnlyOneRelationshipBetweenTwoServers
    {
        check {
            var relations =
                MasterSlaveRelationship.allInstances.select(i|((i.master = self.master and i.slave = self.slave) or
                                                               (i.master = self.slave and i.slave = self.master)) and
                                                               i.master <> i.slave).size();
                if(relations < 2)
                   relations = relations +
                    MasterMasterRelationship.allInstances.select(i|((i.masterSource = self.master and
                                                                     i.masterTarget = self.slave) or
                                                                    (i.masterSource = self.slave and
                                                                     i.masterTarget = self.master)) and
                                                                    i.masterSource <> i.masterTarget).size();
            return (relations <=1);
        }
        message: 'It must be assigned one and only one relationship between two servers'
    }
}
```

Figura 4-51 Código EVL para el Contexto de la Metaclase MasterSlaveRelationship [Elaboración Propia]

La Figura 4-52 muestra el código EVL para el contexto de la metaclase MasterMasterRelationship.

Validations.evl

```
context MasterMasterRelationship
{
    constraint TargetMustHaveOnlyOneMaster
    {
        check {
            var masters = MasterSlaveRelationship.allInstances.select(i|i.slave = self.masterTarget and i.slave <> i.master).size();
            if(masters < 2)
               masters = masters + MasterMasterRelationship.allInstances.select(i|(i.masterSource = self.masterTarget or
                                                i.masterTarget = self.masterTarget) and i.masterSource <> i.masterTarget).size();
            return masters <= 1;
        }
        message: 'The server (' + self.masterTarget.host + ' : ' + self.masterTarget.port + ') must have one and only one master'
    }
    constraint SourceMustHaveOnlyOneMaster
    {
        check {
            var masters = MasterSlaveRelationship.allInstances.select(i|i.slave = self.masterSource and i.slave <> i.master).size();
            if(masters < 2)
               masters = masters + MasterMasterRelationship.allInstances.select(i|(i.masterSource = self.masterSource or
                                               i.masterTarget = self.masterSource) and i.masterSource <> i.masterTarget).size();
            return masters <= 1;
        }
        message: 'The server (' + self.masterSource.host + ' : ' + self.masterSource.port + ') must have one and only one master'
    }
    constraint ExistsOnlyOneRelationshipBetweenTwoServers
    {
        check {
            var relations = MasterSlaveRelationship.allInstances.select(i|((i.master=self.masterSource and i.slave=self.masterTarget) or
                                                                           (i.master=self.masterTarget and i.slave=self.masterSource)) and
                                                                           i.master <> i.slave).size();
                if(relations < 2)
                   relations = relations +
                MasterMasterRelationship.allInstances.select(i|((i.masterSource=self.masterSource and i.masterTarget = self.masterTarget) or
                                                                (i.masterSource=self.masterTarget and i.masterTarget = self.masterSource)) and
                                                                i.masterSource <> i.masterTarget).size();
            return relations <=1;
        }
        message: 'It must be assigned one and only one relationship between two servers'
    }
}
```

Figura 4-52 Código EVL para el Contexto de la Metaclase MasterMasterRelationship [Elaboración Propia]

Luego, en el plugin .custom se ha desarrollado el código Java que invocará al archivo Validations.evl desarrollado, extractos del código fuente en Java se encuentran en el Anexo 1.

Al ejecutar la opción de validación de un modelo de replicación de MySQL se mostrará una pantalla similar al de la Figura 4-53.

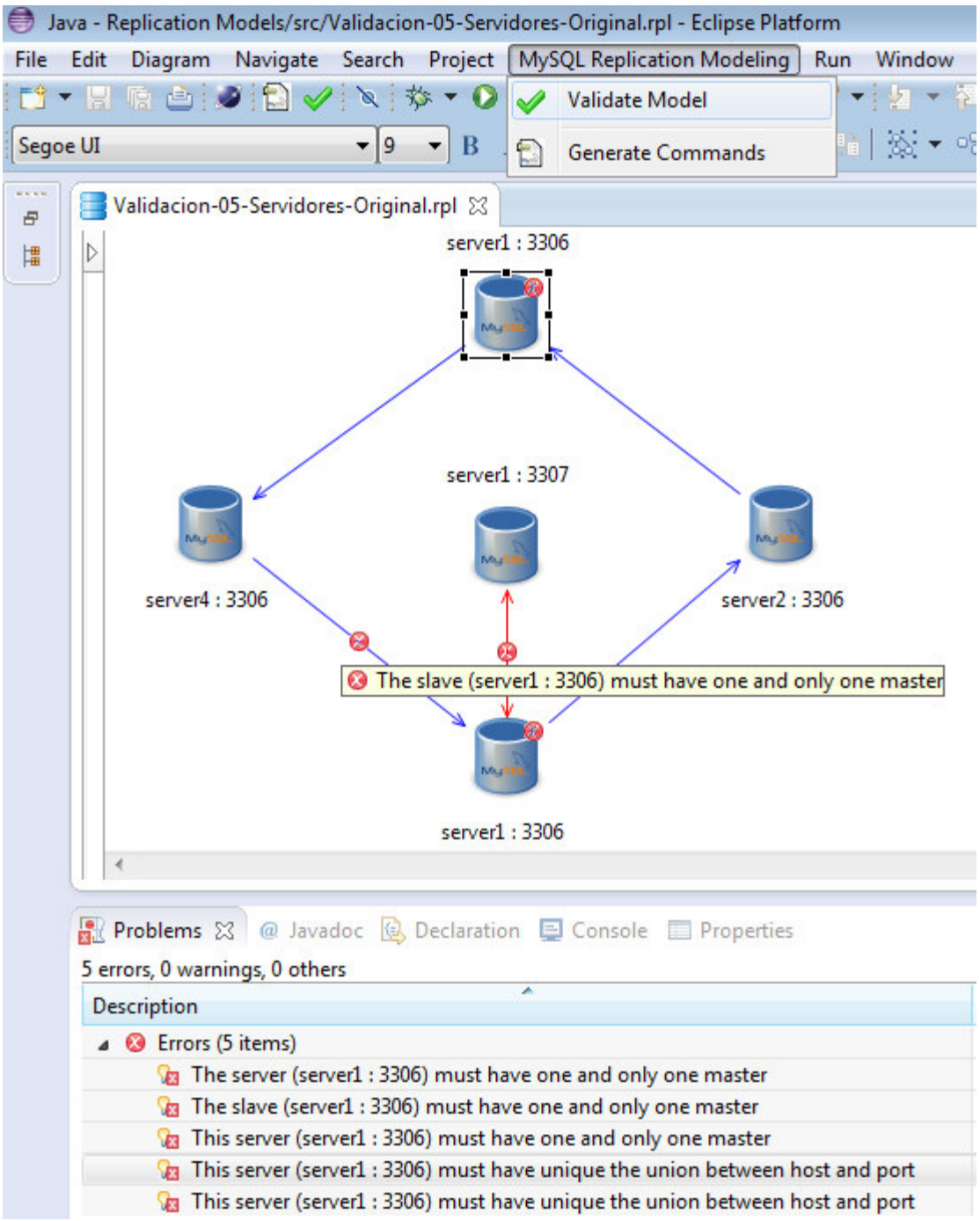


Figura 4-53 Validación de un Modelo de Replicación [Elaboración Propia]

4.4.2.5 Quinta Iteración: Implementación de la Generación de Comandos mysqlreplicate

En esta iteración se va a seguir trabajando en el plugin .custom para desarrollar el código de las transformaciones modelo a texto para generar los comandos de configuración de la replicación de MySQL a partir de un modelo diseñado con la herramienta.

Para el desarrollo de las transformaciones se utilizó el lenguaje Epsilon Generation Language (EGL). EGL es un lenguaje de transformaciones modelo a texto basado en plantillas para la generación de código, documentación y otros artefactos textuales a partir de modelos.

Ahora se va a proceder a agregar los archivos EGL. La Figura 4-54 muestra la pantalla para agregar un archivo EGL.

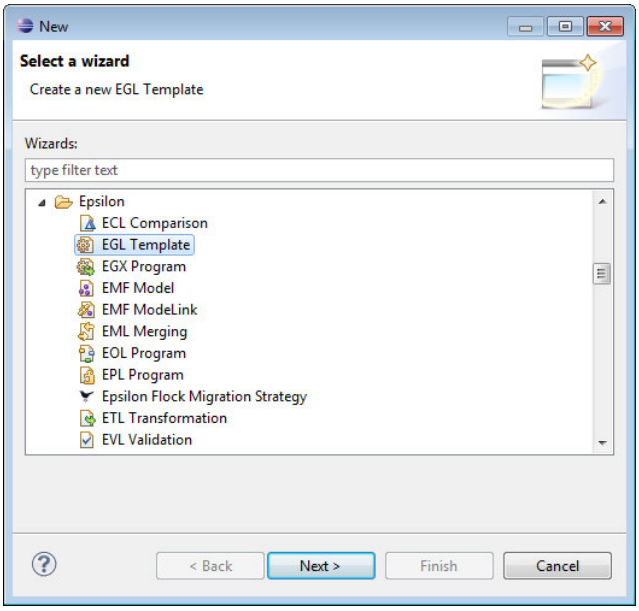


Figura 4-54 Agregar un Archivo EGL [Elaboración Propia]

En la Figura 4-55 se muestra el código EGL desarrollado para generar los comandos mysqlreplicate de configuración a partir de un modelo de replicación.

generation_process.egl

```
[%
TemplateFactory.setTemplateRoot("/egl-templates");
TemplateFactory.setOutputRoot(outputDir);

var tplUtilities : Template = TemplateFactory.load("commands.egl");

tplUtilities.populate("master_slave_links",MasterSlaveRelationship.allInstances);
tplUtilities.populate("master_master_links",MasterMasterRelationship.allInstances);
tplUtilities.process();
tplUtilities.generate(fileName + ".txt");
%]
```

Figura 4-55 Plantilla EGL para Generar Automáticamente los Comandos de Configuración a partir de un Modelo de Replicación [Elaboración Propia]

En la Figura 4-55 se puede observar que se carga el archivo commands.egl, en este archivo se recorrerán todas las relaciones maestro-esclavo y maestro-maestro del modelo de replicación de MySQL. El código de este archivo EGL es mostrado en la Figura 4-56.

commands.egl

```
#*************
#Master->Slave
#*************
[% for(master_slave_link in master_slave_links.sortBy(ms|ms.master.host)) { %]
mysqlreplicate
--master=root@[%=master_slave_link.master.host%]:[%=master_slave_link.master.port%]
--slave=root@[%=master_slave_link.slave.host%]:[%=master_slave_link.slave.port%]
--rpl-user=rpl:rpl -vv --pedantic
[% } %]

#***************
#Master<->Master
#***************
[% for(master_master_link in master_master_links.sortBy(ms|ms.masterSource.host)) { %]
mysqlreplicate
--master=root@[%=master_master_link.masterSource.host%]:[%=master_master_link.masterSource.port%]
--slave=root@[%=master_master_link.masterTarget.host%]:[%=master_master_link.masterTarget.port%]
--rpl-user=rpl:rpl -vv --pedantic
mysqlreplicate
--master=root@[%=master_master_link.masterTarget.host%]:[%=master_master_link.masterTarget.port%]
--slave=root@[%=master_master_link.masterSource.host%]:[%=master_master_link.masterSource.port%]
--rpl-user=rpl:rpl -vv --pedantic
[% } %]
```

Figura 4-56 Archivo commands.egl [Elaboración Propia]

Luego, en el plugin .custom se ha desarrollado el código Java que invocará al archivo generation_process.egl desarrollado, extractos del código fuente en Java se encuentran en el Anexo 1.

Al ejecutar la opción de generación de comandos se mostrará una pantalla similar al de la Figura 4-57.

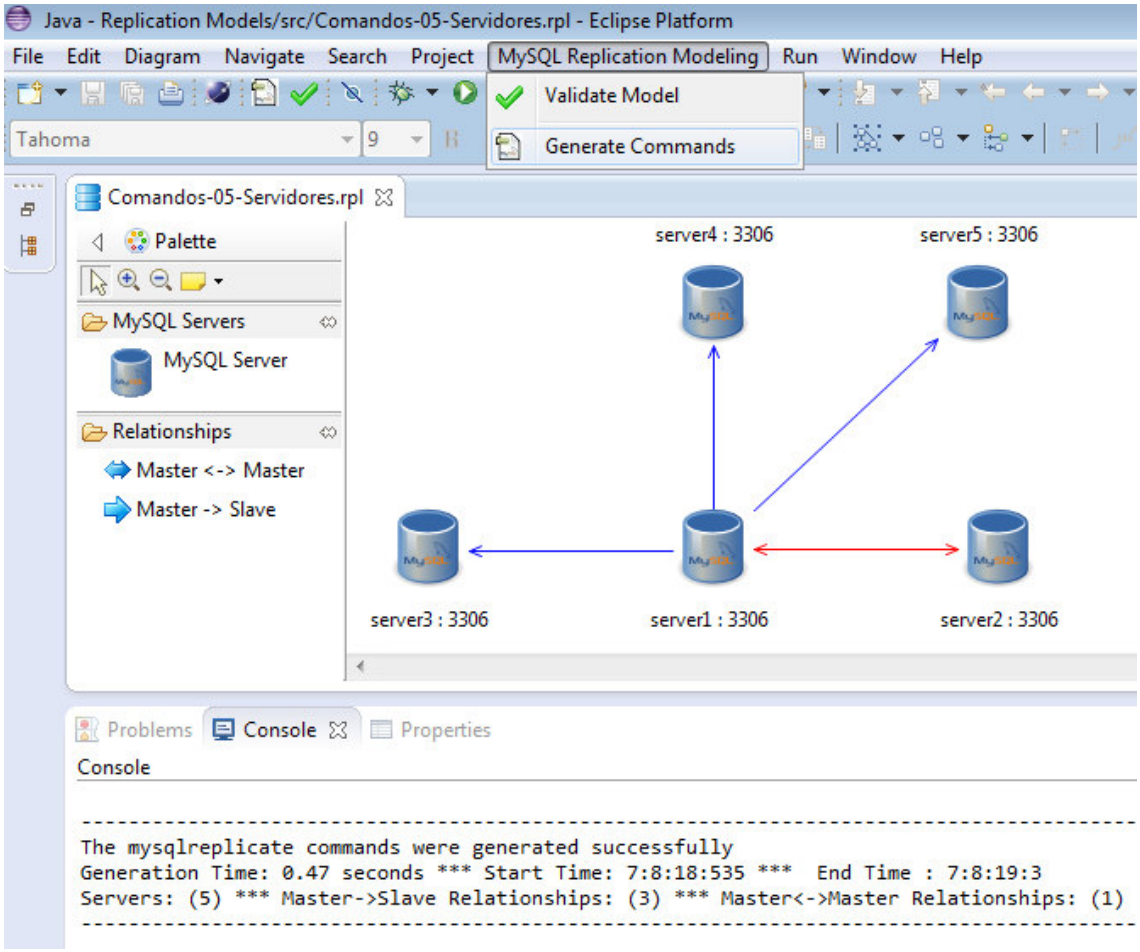


Figura 4-57 Generación de Comandos [Elaboración Propia]

4.4.3 Disciplina de Pruebas

La ventaja de esta herramienta es que está basada en un editor gráfico GMF, que es un componente ya probado, correcto y funcional, y además esta herramienta se integra en el IDE Eclipse por medio de plugins, la cual hereda toda su funcionalidad.

Se realizaron pruebas de funcionalidad propias de un proceso de desarrollo de software para todas las personalizaciones realizadas al editor gráfico GMF.

Esta herramienta será evaluada en el siguiente capítulo por DBA's expertos en el modelado de la replicación de MySQL.

4.5 Fase de Transición

4.5.1 Disciplina de Despliegue

La herramienta se instalará como una serie de plugins dentro del IDE Eclipse, para esto se tiene que generar los archivos .jar de cada plugin, tal como se muestra en la Figura 4-58. Estos plugins (.jars) se copiarán en la carpeta dropins del IDE Eclipse.

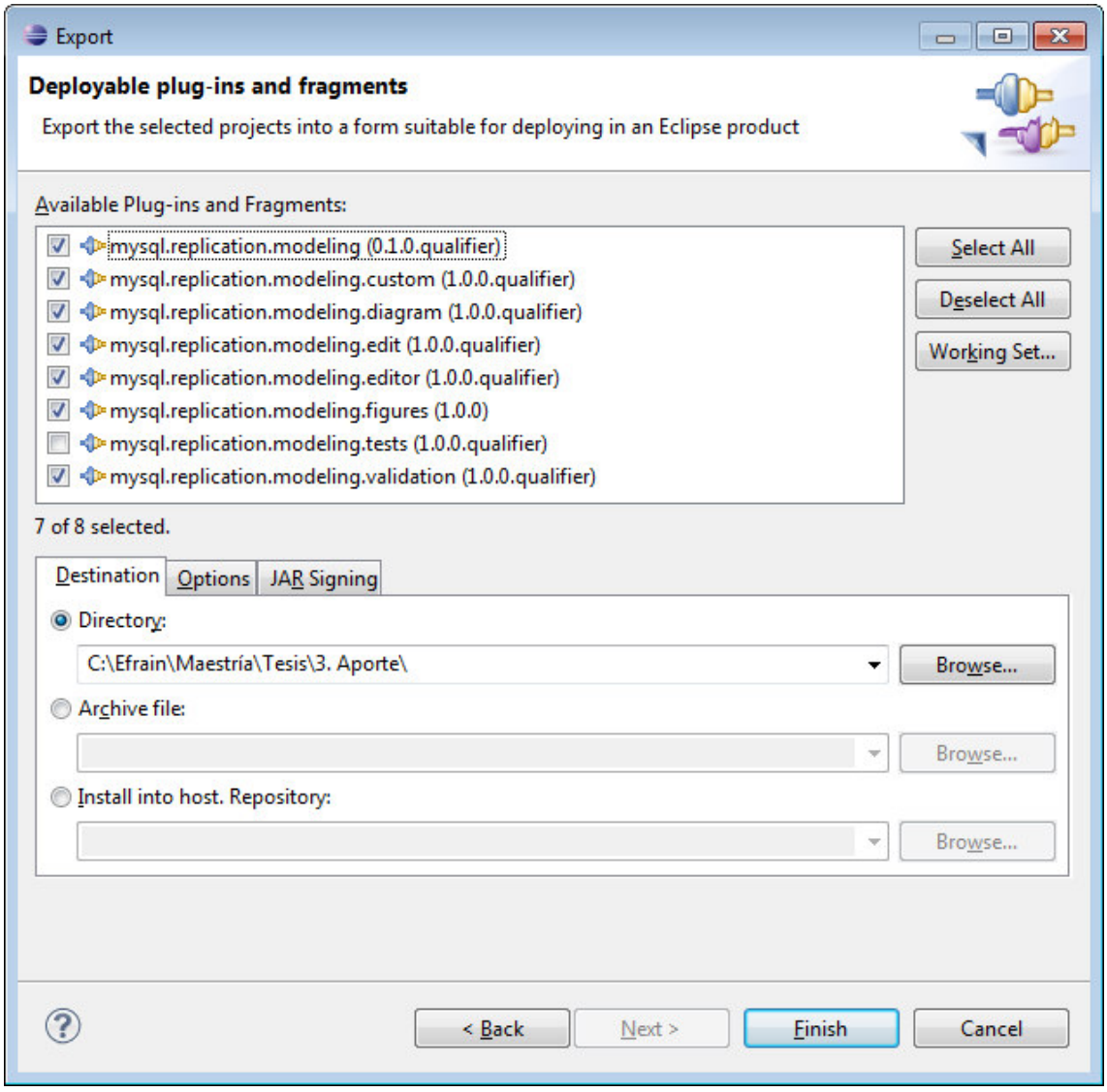


Figura 4-58 Generación de Plugins .jar [Elaboración Propia]

El manual de instalación de la herramienta puede ser consultado en el Anexo 2 y el manual de usuario se encuentra en el Anexo 3.

4.6 Resumen del Capítulo

En este capítulo se ha detallado el proceso de desarrollo de la herramienta propuesta siguiendo las fases del Proceso Unificado de Rational (RUP) y basado en la Ingeniería Dirigida por Modelos (MDE).

En la fase de inicio se trabajó en las disciplinas de análisis del dominio y requerimientos. En la disciplina de análisis del dominio se identificaron las restricciones en la replicación de MySQL, así como las topologías soportadas. En la disciplina de requerimientos se describieron los requerimientos funcionales y no funcionales de la herramienta.

En la segunda fase de elaboración se definió el metamodelo del dominio y se realizó el análisis y diseño de la herramienta basado en las vistas arquitectónicas del documento de arquitectura de software (SAD) del RUP en la que se hizo énfasis en la arquitectura de los frameworks Eclipse Modeling Framework (EMF) y Graphical Modeling Framework (GMF) que fueron seleccionados para el desarrollo de la herramienta bajo la plataforma Eclipse.

En la fase de construcción se trabajó en la disciplina de implementación y pruebas. La implementación se dividió en cinco iteraciones. En la primera iteración se realizó el desarrollo del archivo .emf (Emfatic) del metamodelo. En la segunda iteración se utilizó la herramienta Eugenia para la generación automática del editor gráfico GMF a partir del archivo .emf del metamodelo. En la tercera iteración se desarrollaron las personalizaciones al editor gráfico GMF generado por Eugenia. En la cuarta iteración se desarrollaron las validaciones al metamodelo con el fin de poder determinar si un modelo diseñado con la herramienta es correcto, para ello se utilizó el lenguaje Epsilon Validation Language (EVL). En la quinta y última iteración se desarrolló la generación de los comandos mysqlreplicate de configuración a partir de un modelo de replicación de MySQL válido, para ello se utilizó el lenguaje Epsilon Generation Language (EGL). En la disciplina de pruebas se realizaron todas las pruebas funcionales de la herramienta.

En la fase de transición se trabajó en la disciplina de despliegue donde se describe cómo generar los plugins de la herramienta.

En el siguiente capítulo se describen los experimentos y resultados obtenidos.

Capítulo 5: Experimentos y Resultados

En este capítulo se detalla los experimentos realizados con el fin de validar la herramienta desarrollada y confirmar el cumplimiento del objetivo de la tesis. En la primera sección se describe el diseño de los experimentos. Las instancias de prueba se describen en la segunda sección. La comparación y análisis de los resultados se muestra en la tercera sección y finalmente se realiza un resumen del capítulo. En el siguiente capítulo se presentan las conclusiones de la investigación y los trabajos futuros.

5.1 Diseño de los Experimentos

- Los experimentos se llevaron a cabo con la participación de diez (10) profesionales con experiencia en la replicación MySQL.
- A cada participante se le asignó una PC con Windows 7 Home Premium, procesador Intel Core i5 2.3 GHz, 4GB de RAM, y la instalación de la herramienta propuesta MySQL Replication Modeling y Microsoft Visio 2013.
- Antes de iniciar los experimentos, los participantes fueron capacitados en el uso de ambas herramientas y en las mediciones de tiempo a ser realizadas.
- Los participantes confirmaron el cumplimiento de la funcionalidad de la herramienta propuesta.
- Los experimentos consisten en medir el tiempo empleado en:
    - La corrección por parte del participante, de un modelo de replicación MySQL diseñado en Microsoft Visio 2013. Esta tarea implica la identificación manual de los errores del modelo.
    - La corrección por parte del participante, de un modelo de replicación MySQL diseñado en MySQL Replication Modeling. Esta tarea implica la identificación automática de los errores del modelo y confirmación automática cuando un modelo es válido.
- También se midió el tiempo empleado en:
    - La escritura manual de los comandos mysqlreplicate de un modelo de replicación válido diseñado en Microsoft Visio 2013.
    - La generación automática de los comandos mysqlreplicate a partir de un modelo de replicación válido diseñado en MySQL Replication Modeling.

5.2 Instancias de Prueba

Se definió cinco instancias de prueba conformadas por el número de servidores en un modelo de replicación MySQL, concretamente se trabajó con modelos de 5, 10, 15, 20 y 25 servidores MySQL. En la Tabla 5-1 se muestra el detalle de cada instancia de prueba para la corrección de errores de un modelo de replicación MySQL. En el Anexo 4 se encuentran los modelos de replicación de cada instancia de prueba para ambas herramientas, Microsoft Visio 2013 y MySQL Replication Modeling.

Nombre de Instancia	Servidores MySQL del Modelo	Relaciones Maestro-Esclavo	Relaciones Maestro-Maestro	Errores del Modelo
C1	5	4	1	5
C2	10	8	1	8
C3	15	13	2	8
C4	20	16	3	8
C5	25	24	1	10

Tabla 5-1 Instancias de Prueba para la Corrección de Errores de un Modelo de Replicación MySQL [Elaboración Propia]

En la Tabla 5-2 se muestra el detalle de cada instancia de prueba para la generación de comandos mysqlreplicate a partir de un modelo de replicación MySQL válido. En el Anexo 5 se encuentran los modelos de replicación de cada instancia de prueba para ambas herramientas, Microsoft Visio 2013 y MySQL Replication Modeling.

Nombre de Instancia	Servidores MySQL del Modelo	Relaciones Maestro-Esclavo	Relaciones Maestro-Maestro	Errores del Modelo
G1	5	3	1	0
G2	10	8	1	0
G3	15	13	1	0
G4	20	18	1	0
G5	25	23	1	0

Tabla 5-2 Instancias de Prueba para la Generación de Comandos mysqlreplicate de un Modelo de Replicación MySQL válido [Elaboración Propia]

Los ejemplos de la aplicación de ambas herramientas para las instancias de prueba para la corrección de un modelo de replicación MySQL se encuentran en el Anexo 6 y en el Anexo 7 se encuentran los ejemplos de la aplicación de ambas herramientas para las instancias de prueba para la generación de comandos mysqlreplicate.

5.3 Resultados

Corrección de un Modelo de Replicación MySQL

La Tabla 5-3 muestra los resultados de tiempo promedio empleado por los participantes en la corrección de errores de un modelo de replicación MySQL, al usar Microsoft Visio 2013 y MySQL Replication Modeling.

En los resultados de la Tabla 5-3, se puede observar que el porcentaje de error es 0% al usar ambas herramientas. Sin embargo, para la corrección de errores de un modelo de replicación con 5 servidores, se demuestra que al usar la herramienta propuesta MySQL Replication Modeling, el tiempo empleado se reduce en 62.69%, comparado con el uso de la herramienta Microsoft Visio 2013. Se puede notar en la Tabla 5-3 que a medida que el número de servidores MySQL del modelo de replicación es mayor, la reducción del tiempo aumenta. Para la corrección de errores de un modelo de replicación con 25 servidores, se demuestra que al usar MySQL Replication Modeling, el tiempo empleado se reduce en 87.43%, comparado con el uso de la herramienta Microsoft Visio 2013.

Servidores del Modelo	% Error		Tiempo Empleado (Segundos)		Reducción de Tiempo	
	Microsoft Visio 2013	MySQL Replication Modeling	Microsoft Visio 2013	MySQL Replication Modeling	Segundos	%
5	0%	0%	92.20	34.40	57.80	62.69%
10	0%	0%	193.00	49.30	143.70	74.46%
15	0%	0%	331.00	51.30	279.70	84.50%
20	0%	0%	421.10	53.70	367.40	87.25%
25	0%	0%	595.70	74.90	520.80	87.43%

Tabla 5-3 Tiempo Empleado en la Corrección de un Modelo de Replicación MySQL por MySQL Replication Modeling y Microsoft Visio 2013 [Elaboración Propia]

En el Anexo 8 se pueden encontrar los resultados de los tiempos empleados por cada participante al usar la herramienta Microsoft Visio 2013 y la herramienta propuesta MySQL Replication Modeling, tanto para la corrección de un modelo de replicación MySQL, así como para la generación de comandos mysqlreplicate de configuración, a partir de un modelo de replicación MySQL válido.

En la Figura 5-1 se ilustra usando los datos de la Tabla 5-3, el comportamiento del tiempo empleado en la corrección de errores de un modelo de replicación MySQL, al usar la herramienta propuesta MySQL Replication Modeling, en comparación con la herramienta Microsoft Visio 2013. En la Figura 5-1, se puede observar el área sombreada de la reducción del tiempo, mostrando el porcentaje promedio reducido, así como la reducción del tiempo promedio en minutos.

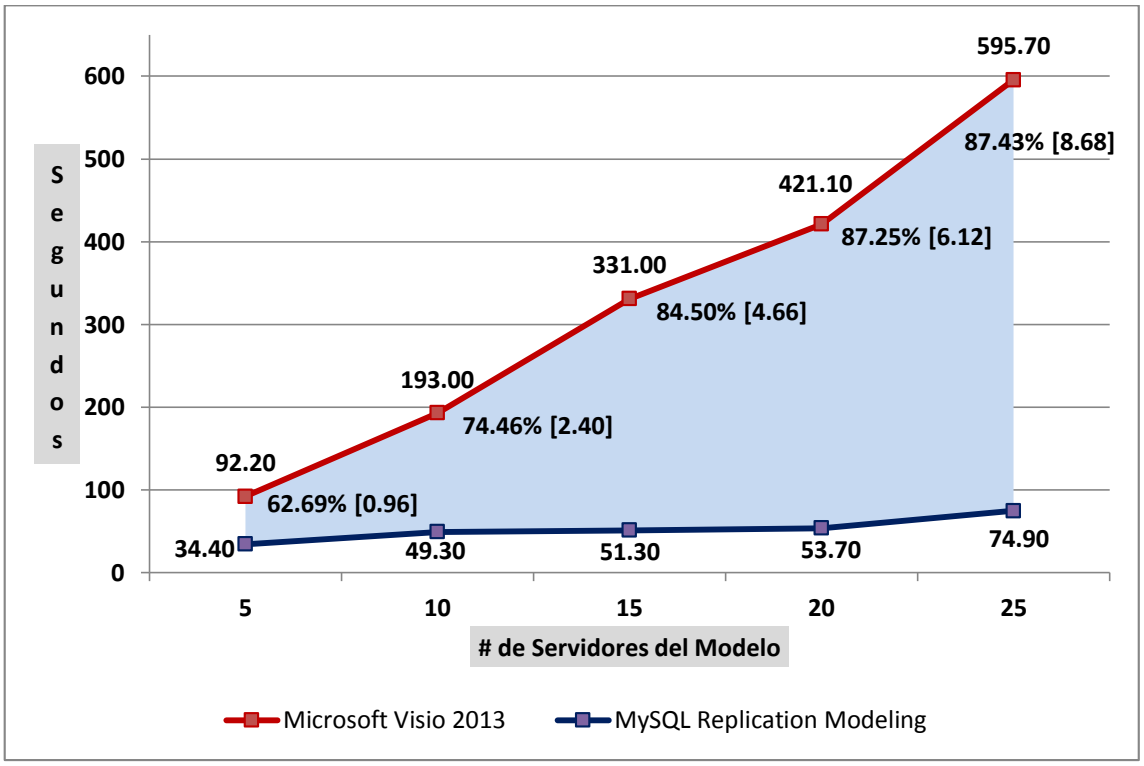


Figura 5-1 Comportamiento del Tiempo en la Corrección de un Modelo de Replicación MySQL de MySQL Replication Modeling y Microsoft Visio 2013 para instancias de la Tabla 5-3 [Elaboración Propia]

Generación de Comandos mysqlreplicate a partir de un Modelo de Replicación

En la Tabla 5-4 se muestra el tiempo promedio empleado por los participantes en la generación de los comandos mysqlreplicate de configuración a partir de un modelo de replicación MySQL válido, al usar Microsoft Visio 2013 y MySQL Replication Modeling.

En los resultados de la Tabla 5-4, se puede observar que el porcentaje de error es 0% al usar ambas herramientas. Sin embargo, para la generación de comandos mysqlreplicate de un modelo de replicación con 5 servidores MySQL, se demuestra que al usar la herramienta propuesta MySQL Replication Modeling, el tiempo empleado se reduce en 99.04%, comparado con el uso de la herramienta Microsoft Visio 2013. Se puede notar en la Tabla 5-4 que a medida que el número de servidores MySQL del modelo de replicación es mayor, la reducción del tiempo aumenta. Para la generación de comandos mysqlreplicate de un modelo de replicación con 25 servidores, se demuestra que al usar MySQL Replication Modeling, el tiempo empleado se reduce en 99.78%, comparado con el uso de la herramienta Microsoft Visio 2013.

Servidores del Modelo	**% Error**		**Tiempo Empleado (Segundos)**		**Reducción de Tiempo**	
	Microsoft Visio 2013	**MySQL Replication Modeling**	**Microsoft Visio 2013**	**MySQL Replication Modeling**	**Segundos**	**%**
5	0%	0%	82.00	0.79	81.21	99.04%
10	0%	0%	161.50	0.81	160.69	99.50%
15	0%	0%	241.00	0.83	240.17	99.66%
20	0%	0%	294.20	0.86	293.34	99.71%
25	0%	0%	397.60	0.88	396.72	99.78%

Tabla 5-4 Tiempo Empleado en la Generación de Comandos mysqlreplicate por MySQL Replication Modeling y Microsoft Visio 2013 [Elaboración Propia]

En la Figura 5-2 se muestra usando los datos de la Tabla 5-4, el comportamiento del tiempo empleado en la generación de comandos mysqlreplicate de configuración a partir de un modelo de replicación válido, al usar la herramienta propuesta MySQL Replication Modeling, en comparación con la herramienta Microsoft Visio 2013. En la Figura 5-2, se puede observar el área sombreada de la reducción del tiempo, mostrando el porcentaje promedio reducido, así como la reducción del tiempo promedio en minutos.

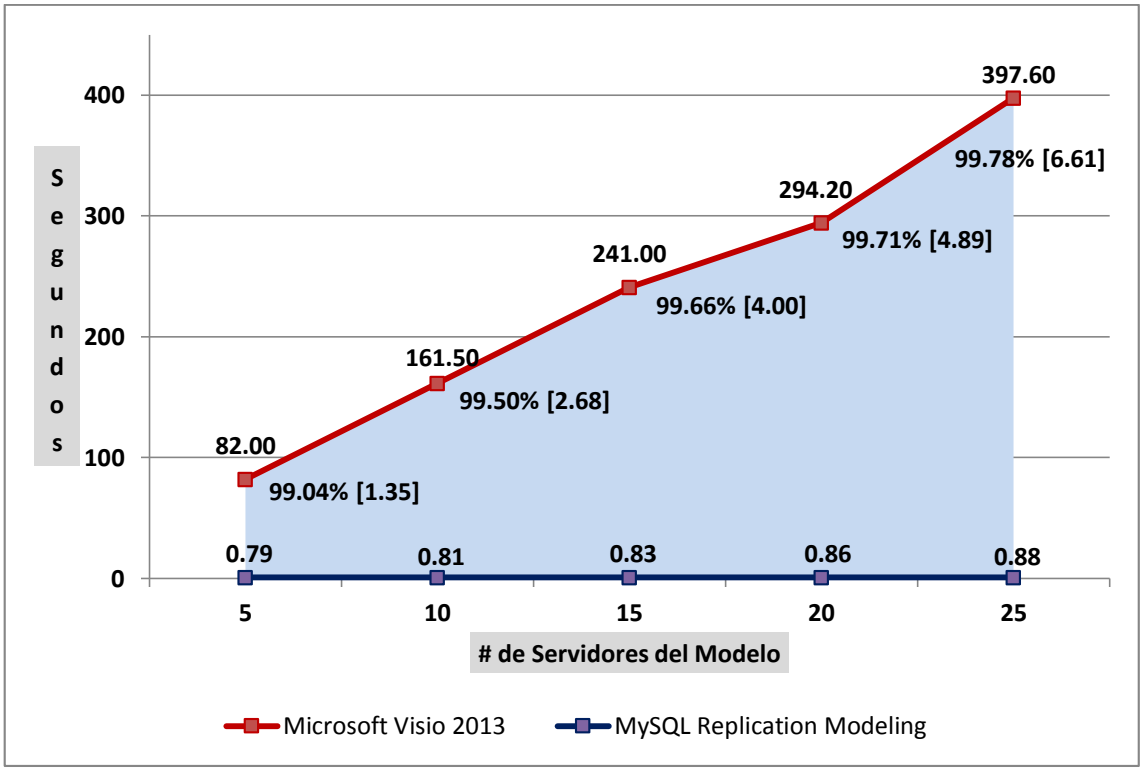


Figura 5-2 Comportamiento del Tiempo en la Generación de Comandos mysqlreplicate de MySQL Replication Modeling y Microsoft Visio 2013 para instancias de la Tabla 5-4 [Elaboración Propia]

5.4 Resumen del Capítulo

Este capítulo se describió los experimentos realizados con el fin de validar la herramienta desarrollada y confirmar el cumplimiento del objetivo de la tesis.

En la primera sección del capítulo se describió el diseño de los experimentos, el hardware y software utilizado.

En la segunda sección, se describieron las instancias de prueba utilizadas, la cual están conformadas por el número de servidores MySQL en un modelo de replicación.

La comparación y análisis de los resultados se mostraron en la tercera sección, en la que se demuestra el beneficio de reducción de tiempo en la corrección de errores de un modelo de replicación MySQL, y en la generación de comandos mysqlreplicate de configuración, al usar la herramienta propuesta MySQL Replication Modeling en comparación con Microsoft Visio 2013.

En el siguiente capítulo se presentan las conclusiones de la investigación y los trabajos futuros.

Capítulo 6: Conclusiones y Trabajos Futuros

6.1 Conclusiones

- Los resultados de los experimentos demostraron que la herramienta propuesta cumple con el objetivo de este trabajo de tesis de investigación, concretamente, con el uso de la herramienta propuesta MySQL Replication Modeling se logró:
  - ✓ Validar automáticamente un Modelo de Replicación MySQL.
  - ✓ Reducir en un 62.69% el tiempo empleado en la corrección de un modelo de replicación MySQL con 5 Servidores, usando Microsoft Visio 2013.
  - ✓ Reducir en un 87.43% el tiempo empleado en la corrección de un modelo de replicación MySQL con 25 servidores, usando Microsoft Visio 2013. La reducción de tiempo será cada vez mayor cuando se incremente el número de servidores en un Modelo de Replicación MySQL.
  - ✓ Generar automáticamente los Comandos mysqlreplicate de Configuración a partir de un Modelo de Replicación MySQL válido.
  - ✓ Reducir en más del 99% el tiempo empleado en la generación de comandos mysqlreplicate a partir de un modelo de replicación MySQL válido, con 5, 10, 15, 20 y 25 servidores usando Microsoft Visio 2013.
- Se estudió la documentación oficial para el Modelado de la Replicación MySQL y se revisaron las herramientas existentes para diagramar modelos de replicación de MySQL, tales como Microsoft Visio [Visio 2013a], Dia [Dia 2013], Draw.io [Draw.io 2013], Gliffy [Gliffy 2014], ConceptDraw [ConceptDraw 2014], Creately [Creately 2014], FreelyDraw [FreelyDraw 2014], SmartDraw [SmartDraw 2014], LucidChart [LucidChart 2014], entre otras.
- Se revisaron artículos científicos sobre el desarrollo de editores gráficos de modelado basados en la Ingeniería Dirigida por Modelos (MDE), en la que se identificó una serie de frameworks, herramientas y lenguajes para MDE, tales como Eclipse Modeling Framework (EMF), Graphical Modeling Framework (GMF), Kybele GMF Generator (KybeleGMFGen), Eugenia, Generic Eclipse Modeling System (GEMS), Generic Modeling Environment (GME), Microsoft Visual Studio Visualization and Modeling SDK (VSVMSDK), Acceleo, Epsilon Validation Language (EVL), Epsilon Generation Language (EGL).

- Se seleccionaron y estudiaron los frameworks EMF y GMF de la plataforma Eclipse, la herramienta Eugenia y los lenguajes EVL y EGL del proyecto Epsilon de Eclipse. Se eligió usar Epsilon dado que es una familia completa de lenguajes y herramientas maduras para desarrollar software basado en MDE. La plataforma Eclipse fue elegida porque es una tecnología madura, software libre y multiplataforma.
- Se desarrolló la herramienta siguiendo las fases del Proceso Unificado de Rational (RUP). Dado que RUP no está orientado al desarrollo de proyectos de investigación, éste fue adaptado y simplificado en este trabajo para un desarrollo basado en la Ingeniería Dirigida por Modelos (MDE). Se realizó el análisis del dominio del modelado de la replicación MySQL, luego se especificaron los requerimientos del software, se definió el metamodelo propuesto y posteriormente se desarrolló el análisis y diseño. Se realizaron cinco iteraciones para la implementación y finalmente se realizaron las pruebas y despliegue de la herramienta.
- Se definió el metamodelo del dominio para el modelado de la replicación MySQL, su codificación se realizó mediante un archivo Emfatic. El metamodelo del dominio y las transformaciones entre modelos son los componentes principales en el desarrollo de la herramienta, así como de todo desarrollo basado en MDE.
- La Ingeniería Dirigida por Modelos (MDE), es un enfoque de la Ingeniería de Software que mediante el uso de frameworks, herramientas y lenguajes, es una técnica adecuada para acelerar el desarrollo de editores gráficos de modelado conocidos en la literatura como Modelado de Dominio Específico (DSM). Los modelos diseñados en un DSM pueden ser validados y posteriormente usados como entrada en un proceso de transformación modelo a modelo y/o modelo a texto, la cual producen automáticamente otros modelos o artefactos basados en texto tales como código fuente o documentación.
- Se contó con la participación de profesionales expertos en la replicación de MySQL para la evaluación de la herramienta propuesta MySQL Replication Modeling, confirmando la identificación automática de errores de un modelo de replicación MySQL y la generación automática de los comandos mysqlreplicate de configuración a partir de un modelo de replicación válido.

6.2 Trabajos Futuros

- Migrar la herramienta a un ambiente web, manteniendo el enfoque basado en la Ingeniería Dirigida por Modelos.
- Trabajar estrechamente con los expertos del dominio de la replicación de MySQL y extender la funcionalidad de la herramienta desarrollada que permita mejorar la productividad de los DBA's y minimizar tareas manuales.
- Extender la funcionalidad de la herramienta para adaptarla al ámbito académico de forma de contar con una herramienta de apoyo para la enseñanza de la replicación de MySQL.
- Desarrollar una herramienta basada en la Ingeniería Dirigida por Modelos para el modelado de la replicación y validación de modelos de replicación de otros sistemas de gestión de base de datos, tales como SQL Server, Oracle, entre otros.
- Integrar las validaciones del metamodelo con una base de reglas de un sistema experto.

Referencias Bibliográficas

[Ahmed+ 2010] M. Ahmed, M.M. Uddin, M.S. Azad, & S. Haseeb. MySQL Performance Analysis on a Limited Resource Server: Fedora vs. Ubuntu Linux. *In Proceedings of the Spring Simulation Multiconference*, SpringSim (2010). (artículo 99).

[Al-Ekram+ 2010] R. Al-Ekram, & R. Holt. OSSR: Optimal Single Site Replication. *In Proceedings of the International Symposium on Parallel and Distributed Processing with Applications*, ISPA (2010). (pp. 433-441).

[Ameller 2009] D. Ameller David. *Considering Non-Functional Requirements in Model-Driven Engineering*. Universidad Politécnica de Cataluña, Barcelona (2009).

[Araújo 2011] M. Araújo. *Database Replication in Large Scale Systems*. Universidad del Miño, Braga (2011).

[Moros 2012] B. Moros. *Un proceso de Ingeniería de Requisitos dirigido por modelos centrado en reutilización.* Universidad de Murcia, Murcia (2012).

[Bovenzi+ 2012] A. Bovenzi, D. Cotroneo, R. Pietrantuono, & S. Russo. On the Aging Effects due to Concurrency Bugs: a Case Study on MySQL. *In International Symposium on Software Reliability Engineering*, ISSRE (2012). (pp. 211-220).

[Bradford+ 2012] R. Bradford, & C. Schneider. *Effective MySQL: Replication Techniques in Depth*. McGraw-Hill (2012).

[Cecchet+ 2008] E. Cecchet, G. Candea, & A. Ailamaki. Middleware-based Database Replication: The Gaps Between Theory and Practice. *In ACM International conference on Management of data*, SIGMOD (2008). (pp. 739-752).

[Cetinkaya+ 2011] D. Cetinkaya, A. Verbraeck, & M.D. Seck. MDD4MS: A Model Driven Development Framework for Modeling and Simulation. *In Summer Computer Simulation Conference*, SCSC (2011). (pp. 113-121).

[ConceptDraw 2014] ConceptDraw. <http://www.conceptdraw.com>, (recuperado 20.02.14).

[Correia+ 2007] A. Correia, J. Pereira, L. Rodrigues, N. Carvalho, & R. Vilaça. GORDA: An Open Architecture for Database Replication. *In IEEE International Symposium on Network Computing and Applications*, NCA (2007). (pp. 287-290).

[Creately 2014] Creately. <http://creately.com>, (recuperado 20.02.14).

[Dantra+ 2009] R. Dantra, J. Grundy, & J. Hosking. A Domain-Specific Visual Language for Report Writing Using Microsoft DSL Tools. *In IEEE Symposium on Visual Languages and Human-Centric Computing*, VL/HCC (2009). (pp. 15-22).

[Dia 2013] Dia. <http://dia-installer.de>, (recuperado 20.02.14).

[Draw.io 2013] Draw.io <https://www.draw.io>, (recuperado 20.02.14).

[Elnikety+ 2005] S. Elnikety, F. Pedone, & W. Zwaenepoel. Database Replication Using Generalized Snapshot Isolation. *In 24th IEEE Symposium on Reliable Distributed Systems*, SRDS (2005). (pp. 73-84).

[Epsilon 2013] Epsilon. <http://www.eclipse.org/epsilon>, (recuperado 20.02.14).

[FreelyDraw 2014] FreelyDraw. <http://freelydraw.com>, (recuperado 20.02.14).

[FromDual 2010] FromDual. MySQL Performance Monitor. <http://www.fromdual.com/mysql-performance-monitor>, (recuperado 22.07.13).

[Gascueña+ 2012] J.M. Gascueña, E. Navarro, & A. Fernández-Caballero. Model-driven engineering techniques for the development of multi-agent systems. *Engineering Applications of Artificial Intelligence* 25 (2012) 159-173.

[Gliffy 2014] Gliffy. <http://www.gliffy.com>, (recuperado 20.02.14).

[Granada+ 2011] D. Granada, J.M. Vara, V. Andrikopoulos, & E. Marcos. Aplicando la ingeniería dirigida por modelos para soportar la evolución de servicios. *Novática* 37(214) (2011) 52-56.

[Jiménez+ 2010] A. Jiménez, J.M. Vara, V. Bollati, & E. Marcos. Mejorando el nivel de automatización en el desarrollo dirigido por modelos de editores gráficos. *Actas de Talleres de las XV Jornadas de Ingeniería del Software y Bases de Datos*, JISBD (2010). (pp. 29-37).

[Jiménez 2012] A. Jiménez. *Incorporando la Gestión de la Trazabilidad en un entorno de Desarrollo de Transformaciones de Modelos Dirigido por Modelos*. Universidad Rey Juan Carlos, Móstoles (2012).

[Kolovos+ 2008] D. Kolovos, R. Paige, & F. Polack. The epsilon transformation language. *In Theory and practice of model transformations,* (2008). (pp. 46-60).

[Kolovos+ 2009a] D. Kolovos, L. Rose, R. Paige, & F. Polack. Raising the Level of Abstraction in the Development of GMF-based Graphical Model Editors. *In Proceedings of the 2009 ICSE Workshop on Modeling in Software Engineering*, MISE (2009). (pp. 13-19).

[Kolovos+ 2009b] D. Kolovos, R. Paige, & F. Polack. On the evolution of OCL for capturing structural constraints in modelling languages. *In Rigorous Methods for Software Construction and Analysis,* (2009). (pp. 204-218).

[Kolovos+ 2010] D. Kolovos, L. Rose, S. Abid, R. Paige, F. Polack & G. Botterweck. Taming EMF and GMF using model transformation. *In Model Driven Engineering Languages and Systems,* (2010). (pp. 211-225).

[Kruchten 2000] P. Kruchten. *The Rational Unified Process: An Introduction*. Addison-Wesley (2000)

[Lau+ 2012] K. Lau, C.M. Tran. X-MAN: An MDE Tool for Component-based System Development. *In 38th Conference on Software Engineering and Advanced Applications*, SEAA (2012). (pp. 158-165).

[Li+ 2013] Y. Li, J. Jun, & L. Ling. Study of Real-time Database Based on Common Information Model. *In Third International Conference on Intelligent System Design and Engineering Applications*, ISDEA (2013). (pp. 1312-1315).

[Lucidchart 2014] Lucidchart. <https://www.lucidchart.com>, (recuperado 20.02.14).

[Mahanta+ 2013] D. Mahanta, M. Ahmed, & U.J. Bora. A study of Bandwidth Management in Computer Networks. *International Journal of Innovative Technology and Exploring Engineering* 2(2) (2013) 69-73.

[Matsunobu 2011] Y. Matsunobu. MHA. <http://code.google.com/p/mysql-master-ha>, (recuperado 20.02.14).

[Visio 2013a] Microsoft Visio. <http://es.wikipedia.org/wiki/Microsoft_Visio>, (recuperado 20.02.14).

[Visio 2013b] Microsoft Visio. <http://office.microsoft.com/en-us/support/getting-started-visio-FX103299477.aspx>, (recuperado 20.02.14).

[Montenegro+ 2011] C.E. Montenegro Marín, P.A. Gaona García, J.M. Cueva Lovelle, & O. Sanjuán Martínez. Aplicación de ingeniería dirigida por modelos (MDA), para la construcción de una herramienta de modelado de dominio específico (DSM) y la creación de módulos en sistemas de gestión de aprendizaje (LMS) independientes de la plataforma, *DYNA* 78(169) (2011) 43-52.

[MySQL 2013a] MySQL. MySQL Editions. <http://www.mysql.com/products>, (recuperado 20.02.14).

[MySQL 2013b] MySQL. MySQL 5.6 Reference Manual. Replication. <http://dev.mysql.com/doc/refman/5.6/en/replication.html>, (recuperado 20.02.14).

[MySQL 2013c] MySQL. MySQL 3.23, 4.0, 4.1 Reference Manual. Introduction to Replication. <http://dev.mysql.com/doc/refman/4.1/en/replication-intro.html>, (recuperado 20.02.14).

[MySQL 2013d] MySQL. MySQL 3.23, 4.0, 4.1 Reference Manual. C.3.46. Changes in Release 3.23.15 (08 May 2000). <http://dev.mysql.com/doc/refman/4.1/en/news-3-23-15.html>, (recuperado 20.02.14).

[MySQL 2013e] MySQL. MySQL 5.6 Replication An Introduction [White Paper]. MySQL (2013), <http://www.mysql.com/why-mysql/white-papers/>, (recuperado 20.02.14).

[MySQL 2013f] MySQL, Why MySQL?. <http://www.mysql.com/why-mysql/>, (recuperado 20.02.14).

[MySQL 2013g] MySQL. MySQL Replication: High Availability Building a Self-Healing Replication Topology [White Paper]. MySQL (2013), <http://www.mysql.com/why-mysql/white-papers/>, (recuperado 20.02.14).

[MySQL 2013h] MySQL. MySQL Enterprise Monitor. <http://www.mysql.com/products/enterprise/monitor.html>, (recuperado 20.02.14).

[MySQL 2013i] MySQL, MySQL 5.6 Replication: Enhancing Scalability and Availability – A Tutorial. <http://www.mysql.com/why-mysql/white-papers/>, (recuperado 20.02.14).

[Noach 2009] S. Noach. Openark Kit. <http://code.openark.org/forge/openark-kit>, (recuperado 20.02.14).

[Papotti+ 2013] P. Papotti, A. do Prado, W. Lopes de Souza, C. Cirilo, & L. Pires. A Quantitative Analysis of Model-Driven Code Generation through Software Experimentation. CAiSE (2013). (pp. 321-337).

[Percona 2011] Percona. Percona Toolkit <http://www.percona.com/software/percona-toolkit>, (recuperado 20.02.14).

[Plattner 2006] C. Plattner. *Ganymed: A Platform for Database Replication*. Escuela Politécnica Federal de Zúrich, Zúrich (2006).

[Quintero+ 2011] J. Quintero, & J. Duitama. Reflexiones acerca de la adopción de enfoques centrados en modelos en el desarrollo de software. *Ingeniería y universidad* 15(1) (2011) 219-243.

[Rodrigues+ 2011] T. Rodrigues, P. Dantas, F.C. Delicato, P.F. Pires, L. Pirmez, T. Batista, C. Miceli, & A. Zomaya. Model-Driven Development of Wireless Sensor Network Applications. *In International Conference on Embedded and Ubiquitous Computing*, EUC (2011). (pp. 11-18).

[Rose+ 2008] L. Rose, R. Paige, D. Kolovos, & F. Polack. The epsilon generation language. *In Model Driven Architecture-Foundations and Applications*, (2008). (pp. 1-16).

[SmartDraw 2014] SmartDraw. <http://www.smartdraw.com>, (recuperado 20.02.14).

[Stahl+ 2006] T. Stahl, & M. Voelter. *Model-Driven Software Development*. Addison-Wesley (2006).

[Spanou 2012] A. Spanou. *Reflective Diagram-Based Model Editing*. Universidad de York, Inglaterra (2012).

[Sybase 2006] Sybase. MDD for Sybase Replication Server [White Paper]. Sybase (2006), <http://www.sybase.com/files/White_Papers/Sybase_RepServerMDD_wp.pdf>, (recuperado 20.02.14).

[Tekinerdogan+ 2013] B. Tekinerdogan, & E. Demirli. Evaluation Framework for Software Architecture Viewpoint Languages. *In International ACM Sigsoft conference on Quality of software architectures*, QoSA (2013). (pp. 89-98).

[Tomic+ 2013] A. Tomic, D. Sciascia, & F. Pedone. MoSQL: An Elastic Storage Engine for MySQL. *In 28th Annual ACM Symposium on Applied Computing*, SAC (2013). (pp. 455-462).

[Voelter, 2009] M. Voelter. Best Practices for DSLs and Model-Driven Development. *Journal of Object Technology* 8(6) (2009) 79-102.

[Whittle+ 2013] J. Whittle, J. Hutchinson, & M. Rouncefield. The State of Practice in Model-Driven Engineering. *Software IEEE* PP(99) (2013).

[Wiesmann+ 2000] M. Wiesmann, F. Pedone, A. Schiper, B. Kemme, & G. Alonso. Database Replication Techniques: a Three Parameter Classification. *In Proceedings of the 19th IEEE Symposium on Reliable Distributed Systems*, SRDS (2000). (pp. 206-215).

Anexos

Anexo 1: Extractos del Código Fuente

La siguiente figura muestra un extracto del código fuente de la clase CustomAction.

CustomAction.java

```
1  package mysql.replication.modeling.custom.view;
2
3  import java.util.Calendar;
15
16 public class CustomAction implements IWorkbenchWindowActionDelegate
17 {
18     private IEditorPart _objEditor;
19     private Mysql_replication_modelingDiagramEditor _objDiagramEditor;
20     private IAction _objAction;
21     private double _time_start = 0;
22     private Calendar _datetime_start;
23     @Override
24     public void run(final IAction action) {
25         try
26         {
27             _time_start = (double) System.currentTimeMillis();
28             _datetime_start = Calendar.getInstance();
29             action.setEnabled(false);
30             _objEditor = PlatformUI.getWorkbench().getActiveWorkbenchWindow().getActivePage().getActiveEditor();
31             if(_objEditor instanceof Mysql_replication_modelingDiagramEditor)
32             {
33                 _objAction = action;
34                 _objDiagramEditor = (Mysql_replication_modelingDiagramEditor) _objEditor;
35                 switch(_objAction.getId())
36                 {
37                     case "validate_model": validateModel(); break;
38                     case "generate_commands": generateCommands(); break;
39                 }
40                 action.setEnabled(true);
41             }
42             else
43             {
44                 MyConsole.writeError(LabelProperties.getValue("not_replication_model"));
45                 action.setEnabled(true);
46             }
47         }
48         catch(Exception ex)
49         {
50             MyLogger.logger.severe(ex.getMessage() + " ** " + MyLogger.getStackTraceAsString(ex));
51             MyConsole.writeError(ex.getMessage() + " ** " + MyLogger.getStackTraceAsString(ex));
52             action.setEnabled(true);
53         }
54     }
55
56     private void validateModel() throws Exception
57     {
58         ModelController objModelController = null;
59         try
60         {
61             objModelController = new ModelController(
62                     this._time_start, this._datetime_start, this._objDiagramEditor);
63             objModelController.validateModel();
64         }
65         catch(Exception ex)
66         {
67             throw ex;
68         }
69         finally
70         {
71             objModelController = null;
72         }
73     }
74
75     private void generateCommands() throws Exception
76     {
77         ModelController objModelController = null;
78         try
79         {
80             objModelController = new ModelController(
81                     this._time_start, this._datetime_start, this._objDiagramEditor);
82             objModelController.generateCommands();
83         }
84         catch(Exception ex)
85         {
86             throw ex;
87         }
88         finally
89         {
90             objModelController = null;
91         }
92     }
```

La siguiente figura muestra un extracto del código fuente de la clase Model.

Model.java

```
package mysql.replication.modeling.custom.model;

import java.util.Calendar;

public class Model
{
    public Resource getModelResource(Mysql_replication_modelingDiagramEditor lpObjDiagramEditor)
    {
        try
        {
            ResourceSet objResourceSet = lpObjDiagramEditor.getEditingDomain().getResourceSet();
            for (Resource objResource : objResourceSet.getResources()) {
                if ((objResource instanceof GMFResource) == true) {
                    return objResource;
                }
            }
        }
        catch(Exception ex)
        {
            throw ex;
        }
        return null;
    }

    public int getNumberOfErrors(Mysql_replication_modelingDiagramEditor lpObjDiagramEditor) throws Exception
    {
        int intErrors = 0;
        int intWarnings = 0;
        try
        {
            Resource objResource = getModelResource(lpObjDiagramEditor);
            if(objResource == null) return 0;
            Collection<EvlUnsatisfiedConstraint> lstUnsatisfiedConstraints = getUnsatisfiedConstraints(objResource);
            if (lstUnsatisfiedConstraints != null && lstUnsatisfiedConstraints.size() > 0)
            {
                for(EvlUnsatisfiedConstraint objConstraint : lstUnsatisfiedConstraints) {
                    EvlConstraint objEvlConstraint = objConstraint.getConstraint();
                    if(objEvlConstraint.isCritique())
                       intWarnings++;
                }
                intErrors = lstUnsatisfiedConstraints.size() - intWarnings;
            }
        }
        catch(Exception ex)
        {
            throw ex;
        }
        return intErrors;
    }

    public Collection<EvlUnsatisfiedConstraint> getUnsatisfiedConstraints(Resource lpObjResource)
            throws Exception
    {
        Collection<EvlUnsatisfiedConstraint> lstUnsatisfiedConstraints = null;
        try
        {
            if(lpObjResource == null) return null;
            InMemoryEmfModel objModel = new InMemoryEmfModel("M", lpObjResource, Mysql_replication_modelingPackage.eINSTANCE);
            EvlModule objEvlModule = new EvlModule();
            objEvlModule.parse(mysql.replication.modeling.validation.Activator.class.getResource("Validations.evl").toURI());
            objEvlModule.getContext().getModelRepository().addModel(objModel);
            objEvlModule.execute();
            lstUnsatisfiedConstraints = objEvlModule.getContext().getUnsatisfiedConstraints();
        }
        catch(Exception ex)
        {
            throw ex;
        }
        return lstUnsatisfiedConstraints;
    }

    public boolean hasRelationships(Mysql_replication_modelingDiagramEditor lpObjDiagramEditor)
    {
        boolean boolHasRelationships = false;
        try
        {
           Resource objResource = getModelResource(lpObjDiagramEditor);
           if(objResource == null) return false;
           TreeIterator<EObject> it = objResource.getAllContents();
           while(it.hasNext()) {
               EObject objCurrent = it.next();
               if(objCurrent instanceof MasterMasterRelationship || objCurrent instanceof MasterSlaveRelationship)
               {
                  boolHasRelationships = true;
                  break;
               }
           }
        }
        catch(Exception ex)
        {
            throw ex;
        }
        return boolHasRelationships;
    }

    public void generateCommands(double time_start, Calendar datetime_start,
            Mysql_replication_modelingDiagramEditor objDiagramEditor) throws Exception
    {
        ModelController objModelController = null;
        try
        {
            objModelController = new ModelController(
                    time_start, datetime_start, objDiagramEditor);
            objModelController.generateCommands();
        }
        catch(Exception ex)
        {
            throw ex;
        }
        finally
        {
            objModelController = null;
        }
    }
```

La siguiente figura muestra el código fuente del método isValidModel, que nos permite para saber si un modelo de replicación es válido.

```
public boolean isValidModel() throws Exception
{
    boolean boolIsValid = false;
    try
    {
        int intServers = this.getNumberOfServers();
        if(intServers > 0)
        {
            int intErrors = this.getNumberOfErrors(this._objDiagramEditor);
            if(intErrors >  0)
            {
               this.validate();
               String message = getValidationLogMessage(MessageFormat.format(LabelProperties.getValue("model_errors"), intErrors));
               MyConsole.writeError(message);
            }
            else
            {
                this.validate();
                boolean boolHasRelationships = this.hasRelationships(this._objDiagramEditor);
                if(boolHasRelationships)
                   boolIsValid = true;
                else
                {
                   String message = getValidationLogMessage(LabelProperties.getValue("model_no_relationships"));
                   MyConsole.writeError(message);
                }
            }
        }
        else
        {
            this.validate();
            String message = getValidationLogMessage(LabelProperties.getValue("model_no_servers"));
            MyConsole.writeError(message);
        }
    }
    catch(Exception ex)
    {
        throw ex;
    }
    return boolIsValid;
}
```

La siguiente figura muestra el código fuente del método executeEGL, que permite generar los comandos de configuración.

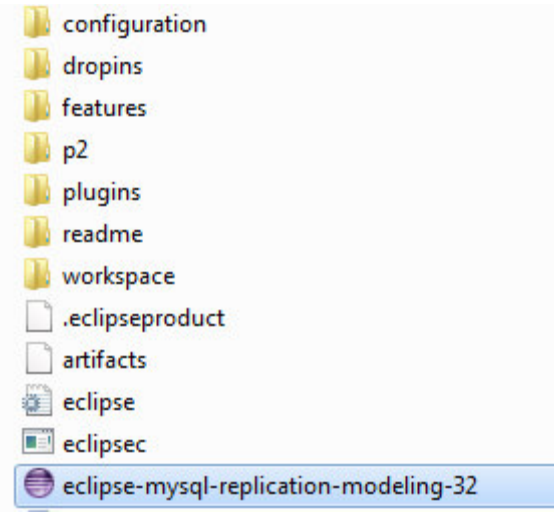
```
public boolean executeEGL(Mysql_replication_modelingDiagramEditor lpObjDiagramEditor,
        String lpStrEGLTemplatePath, String lpStrOutputDir) throws Exception
{
    boolean boolGenerated = false;
    try
    {
        Resource objResource = getModelResource(lpObjDiagramEditor);
        if(objResource == null) return false;
        EglTemplateFactoryModuleAdapter module = new EglTemplateFactoryModuleAdapter(new EglFileGeneratingTemplateFactory());
        java.net.URI objUri = Activator.getDefault().getBundle().getResource(lpStrEGLTemplatePath).toURI();
        module.parse(objUri);
        if(module.getParseProblems() != null && module.getParseProblems().size() > 0) return false;
        module.getContext().getFrameStack().put(Variable.createReadOnlyVariable("outputDir", lpStrOutputDir));
        module.getContext().getFrameStack().put(Variable.createReadOnlyVariable("fileName", lpObjDiagramEditor.getDiagram().getName()));
        InMemoryEmfModel objModel = new InMemoryEmfModel("M", objResource, Mysql_replication_modelingPackage.eINSTANCE);
        module.getContext().getModelRepository().addModel(objModel);
        module.execute();
        module.getContext().getModelRepository().dispose();
        boolGenerated = true;
    }
    catch(Exception ex)
    {
        throw ex;
    }
    return boolGenerated;
}
```

Anexo 2: Manual de Instalación

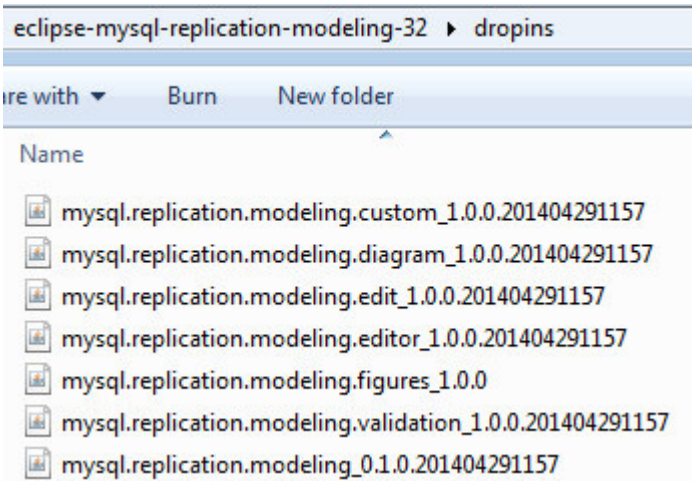
La herramienta se desarrolló y desplegó como plugins (archivos .jar) en el IDE Eclipse Kepler 4.3 para el sistema operativo Windows de 32 bits, este IDE contiene una versión estable de Epsilon (v1.1_SR1), EMF, GMF y Emfatic, solo hace falta tener instalado el JRE 7 o superior de 32 bits.

Requisitos Mínimos del Sistema Windows: Windows XP, 2GB de RAM, JRE 7.

Para empezar a utilizar la herramienta basta con ejecutar el archivo **eclipse-mysql-replication-modeling-32.exe** tal como se muestra en la figura.



La siguiente figura muestra los plugins dentro de la carpeta dropins.

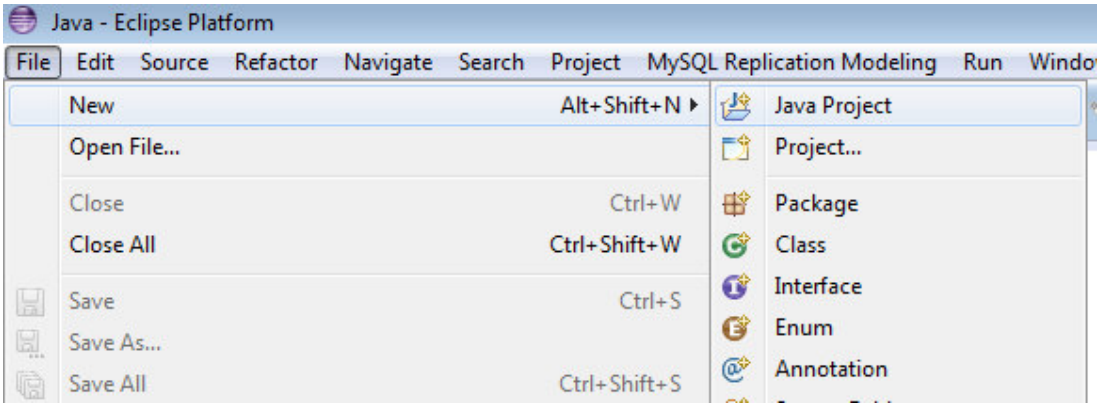


Para generar las versiones para Windows de 64 bits, Linux de 32 y 64 bits y Mac OS de 32 y 64 bits, es necesario portar el código y generar los plugins en el IDE Eclipse Kepler 4.3 para el correspondiente sistema operativo y JRE de 32 o 64 bits, según corresponda.

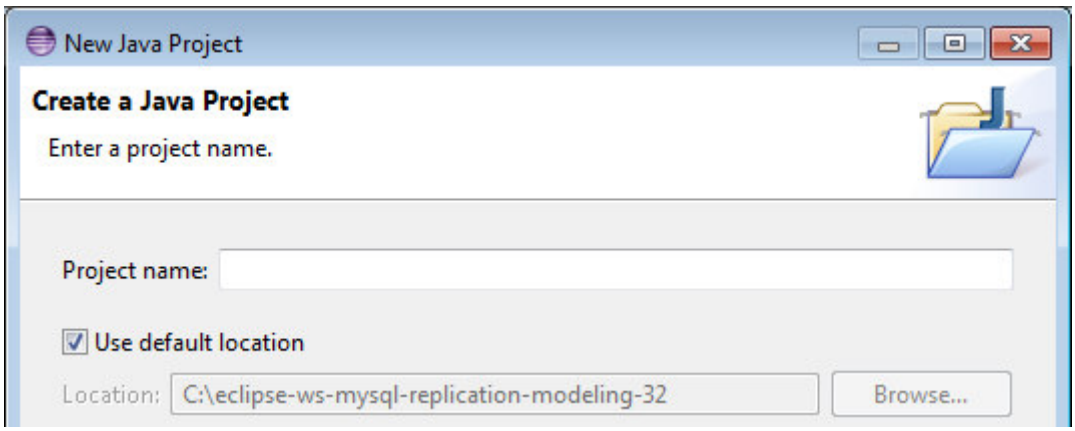
Anexo 3: Manual de Usuario

- **Crear un Proyecto de Java**

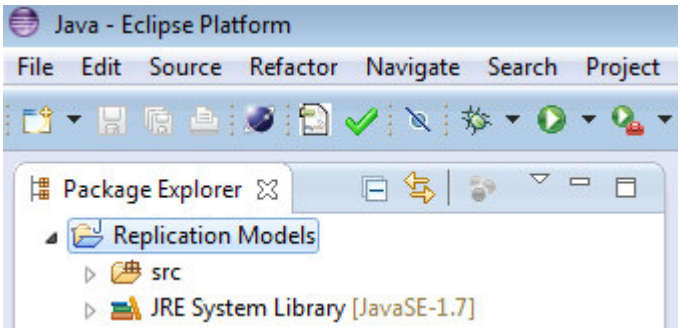
Para poder crear modelos replicación de MySQL, primero se tiene que crear un proyecto Java dando click en el menú **File**, seleccionar **New** y finalmente dar click en la opción Java Project, tal como se muestra en la siguiente figura.



Luego aparecerá una ventana para asignar el nombre del proyecto y confirmar su creación.

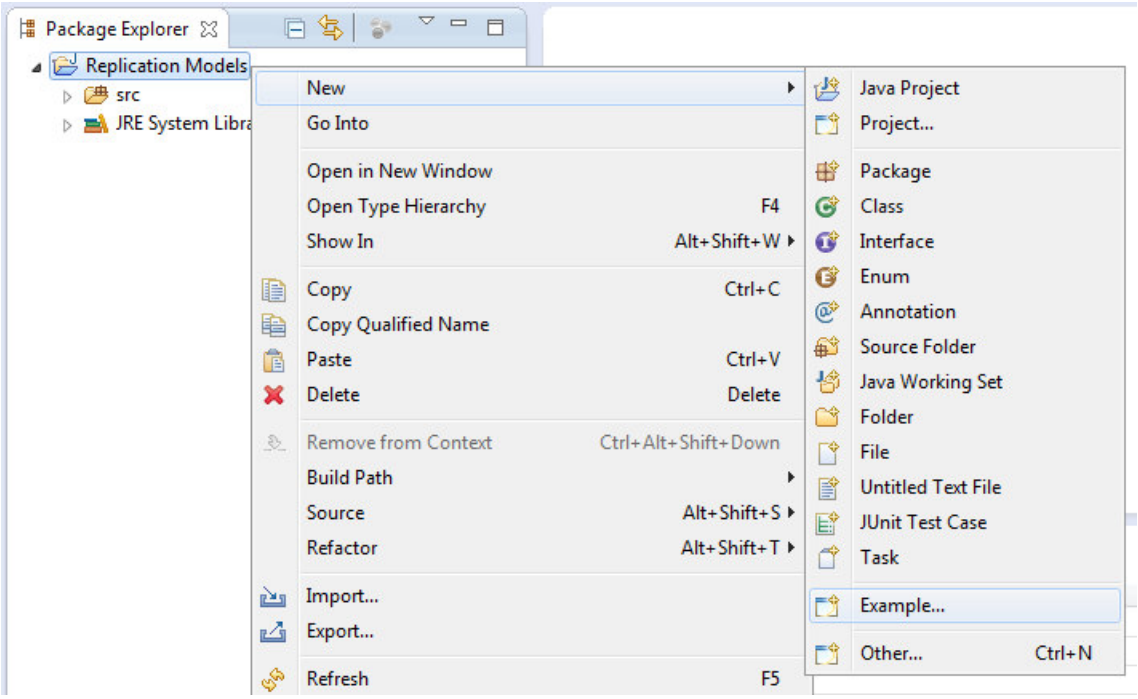


El proyecto se mostrará en el explorador de paquetes, tal como se muestra en la siguiente figura.

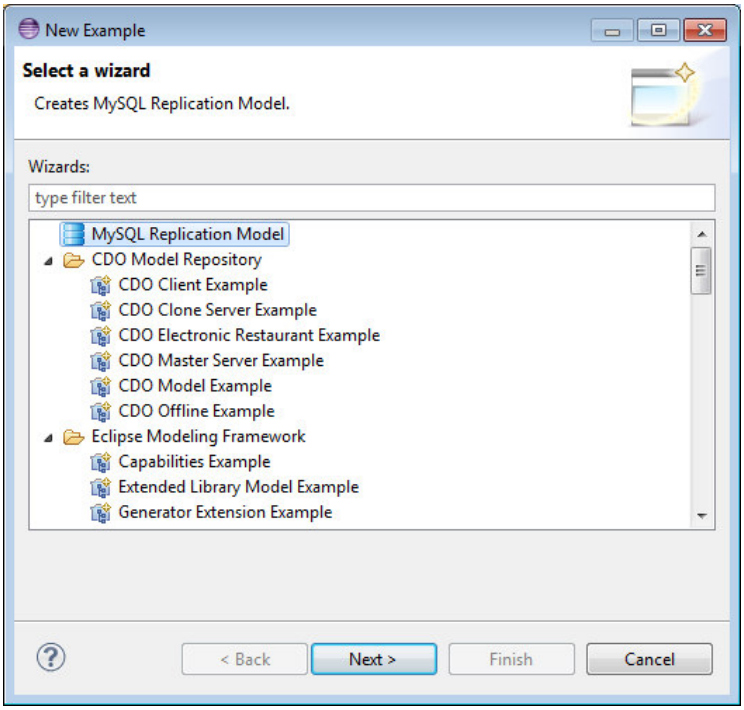


- **Crear un Modelo de Replicación de MySQL**

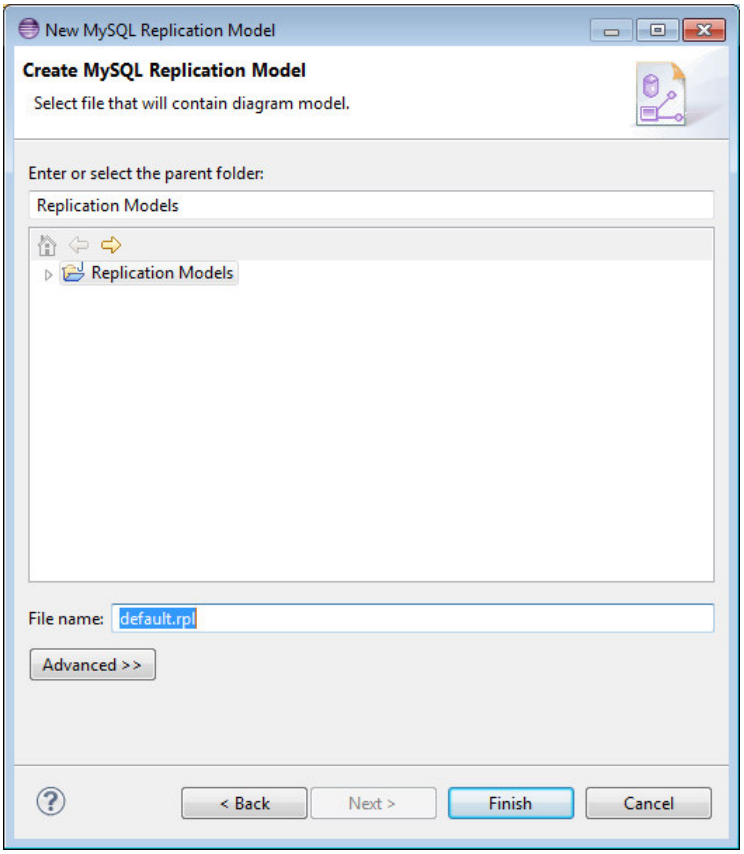
Para crear un modelo de replicación de MySQL, se debe seleccionar y dar click derecho en el proyecto creado anteriormente, click en **New** y finalmente dar click en la opción **Example…** tal como se muestra en la siguiente figura.



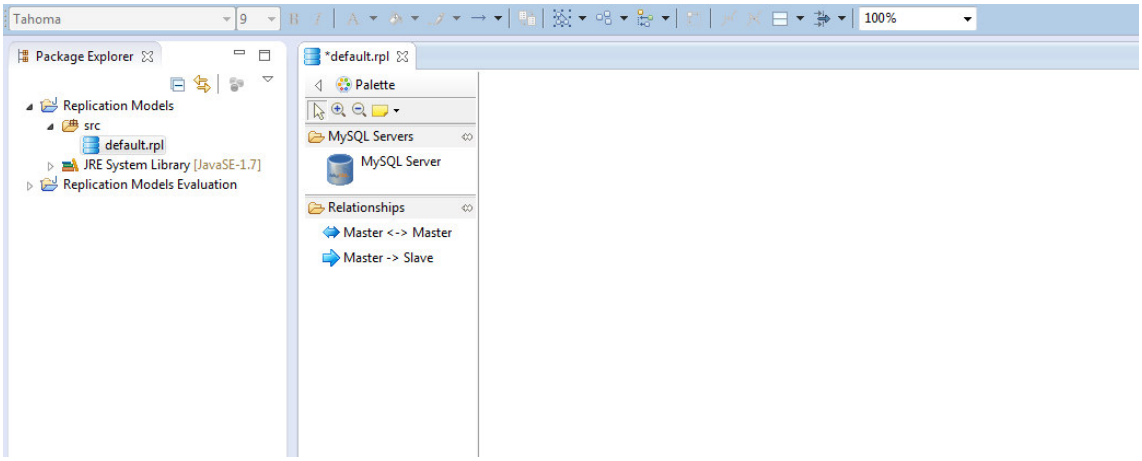
Luego aparecerá la ventana mostrada en la siguiente figura, en la que debemos seleccionar la opción **MySQL Replication Model**.



Se mostrará una nueva ventana para asignar un nombre al modelo de replicación de MySQL, notar que los modelos de replicación tienen extensión .rpl.

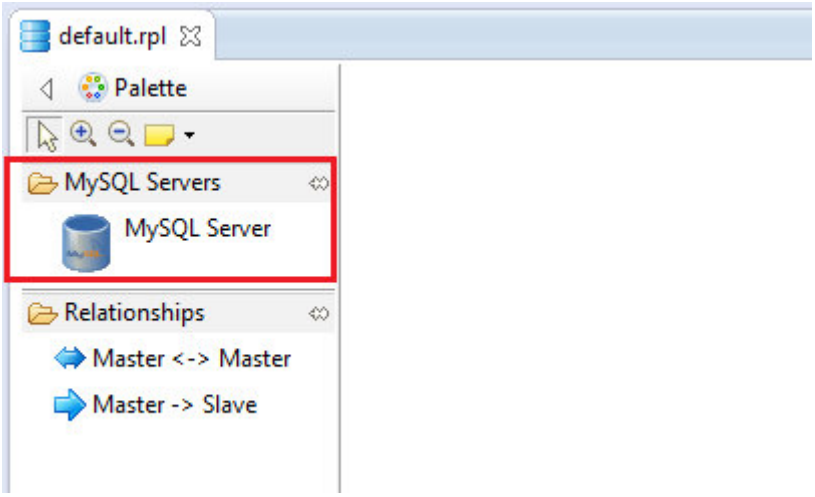


Finalmente, se creará un modelo de replicación vacío dentro de la carpeta src del proyecto, tal como se muestra en la siguiente figura. También podemos observar que se muestra una paleta de herramientas con servidores MySQL y relaciones de replicación, así como también el área para diagramar el modelo de replicación de MySQL.



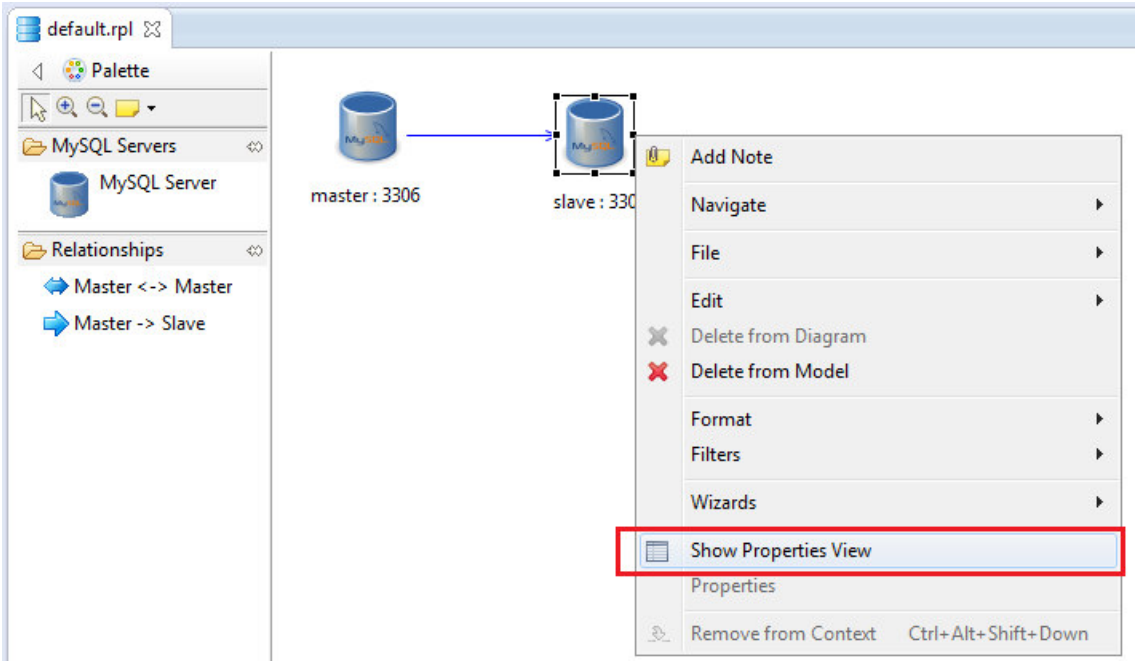
- **Agregar Servidor MySQL al Modelo de Replicación**

Para agregar un servidor MySQL al modelo se debe arrastrar y soltar (drag and drop) un nodo **MySQL Server** de la paleta de herramientas tal como se muestra en la siguiente figura.

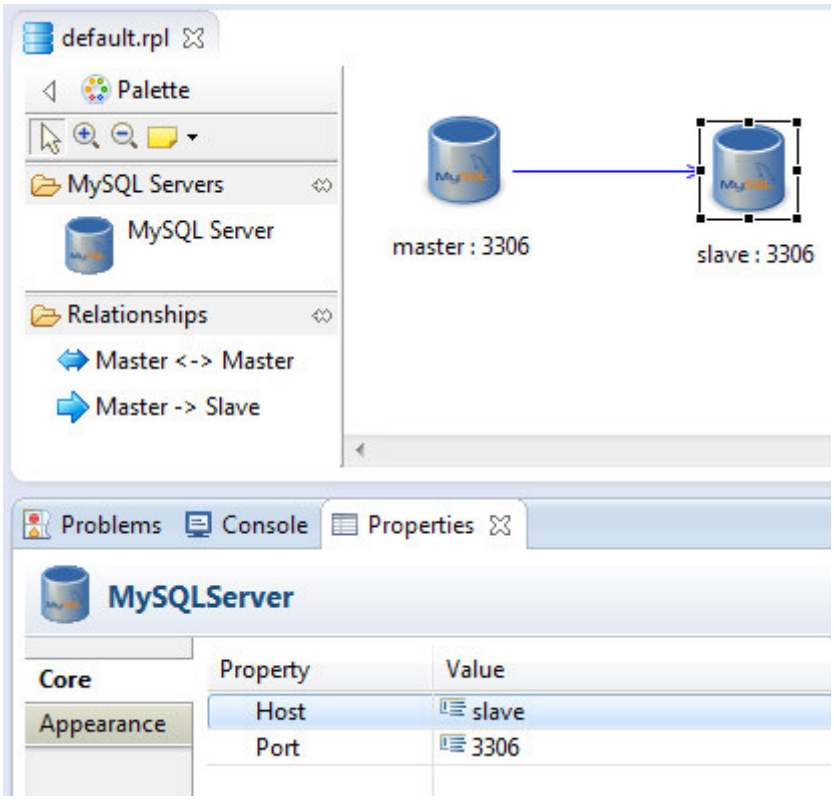


- **Modificar Datos de un Servidor MySQL**

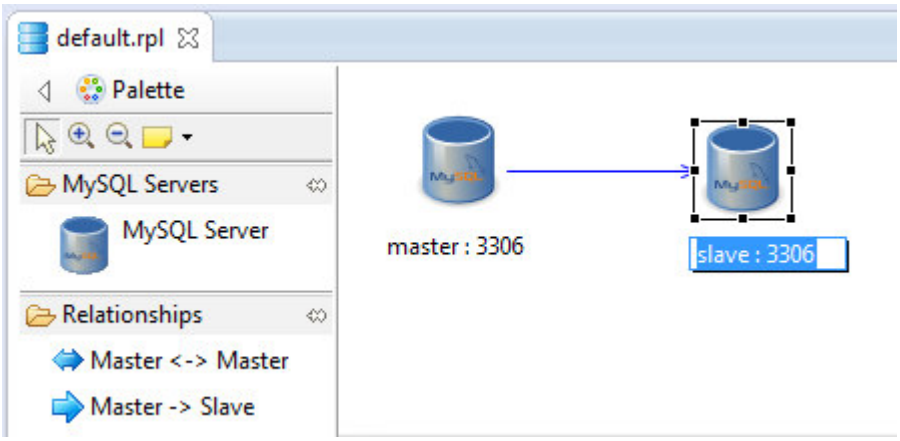
Para modificar los datos de un servidor MySQL del modelo, se debe de seleccionar el servidor MySQL, dar click derecho y luego click en **Show Properties View,** tal como se muestra en la siguiente figura.



Luego se mostrará la vista de propiedades tal como se muestra en la siguiente figura.

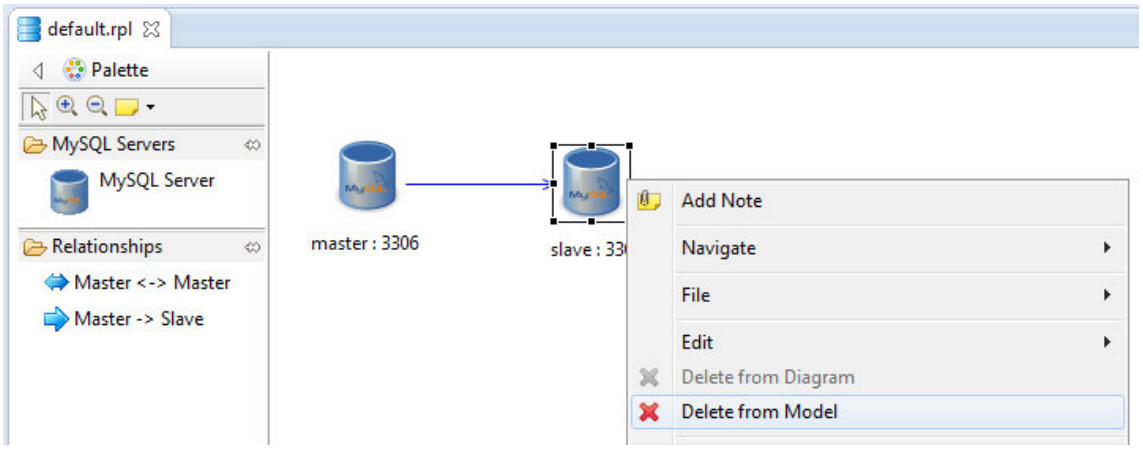


También se puede modificar los datos de un servidor MySQL, seleccionándolo y presionando la tecla F2, tal como se puede observar en la siguiente figura.



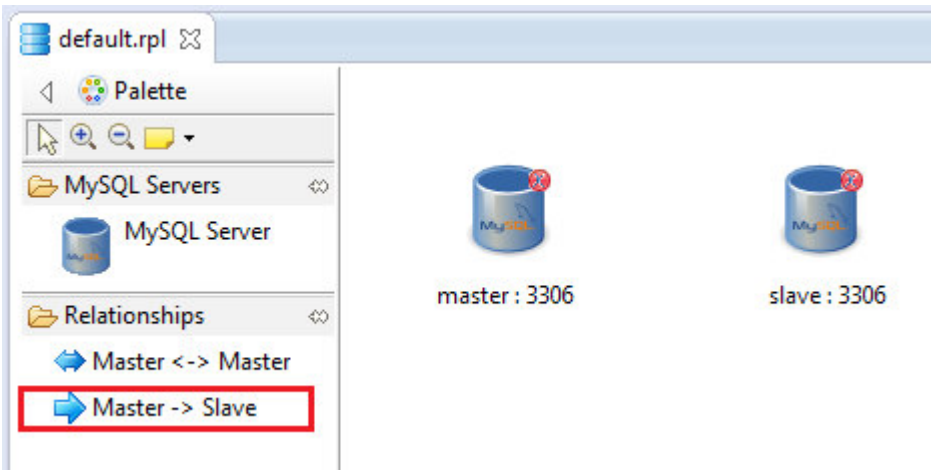
- **Eliminar Servidor MySQL del Modelo**

Para eliminar un servidor MySQL del modelo, basta con seleccionar el servidor MySQL y presionar la tecla suprimir. También se puede eliminar un servidor, seleccionándolo y dando click derecho sobre él y finalmente dando click en la opción **Delete from Model** tal como se puede observar en la siguiente figura.

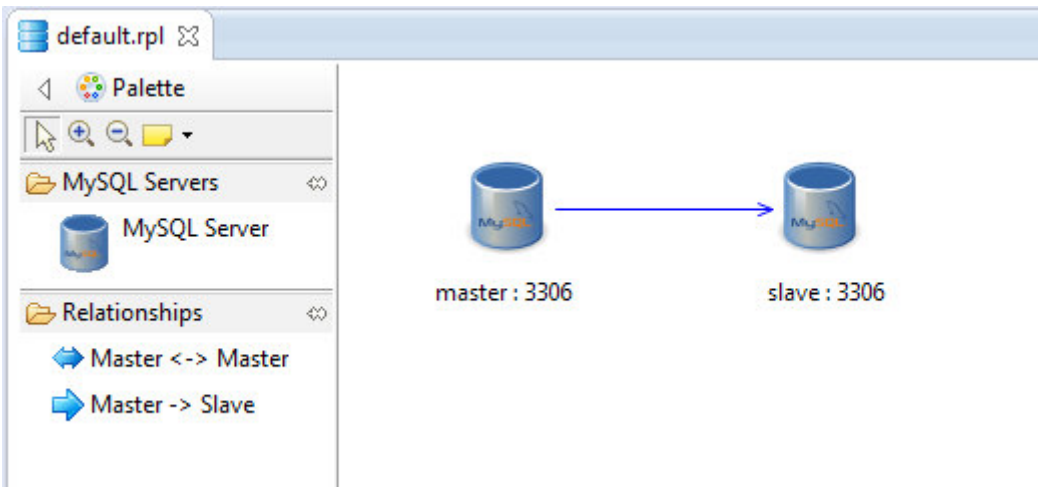


- **Agregar Relación Maestro-Esclavo**

Para agregar una relación maestro-esclavo al modelo se debe arrastrar y soltar (drag and drop) una relación Master - > Slave de la paleta de herramientas, tal como se muestra en la siguiente figura.

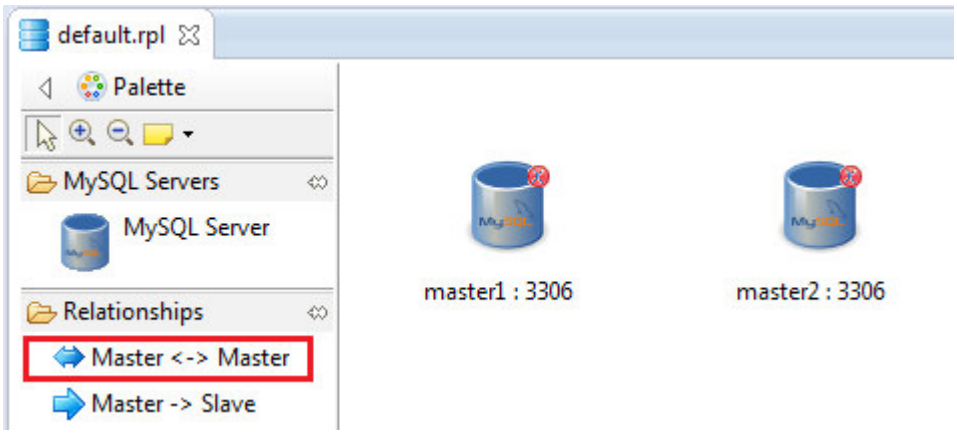


Luego de crear la relación maestro-esclavo en el modelo, éste debe lucir como la imagen de la siguiente figura.

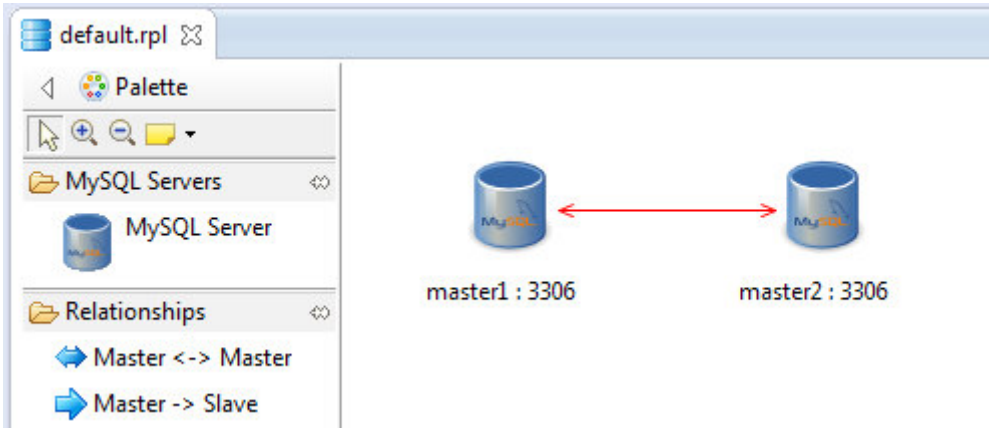


- **Agregar Relación Maestro-Maestro**

Para agregar una relación maestro-maestro, se debe arrastrar y soltar una relación Master < - > Master de la paleta de herramientas, tal como se muestra en la figura.

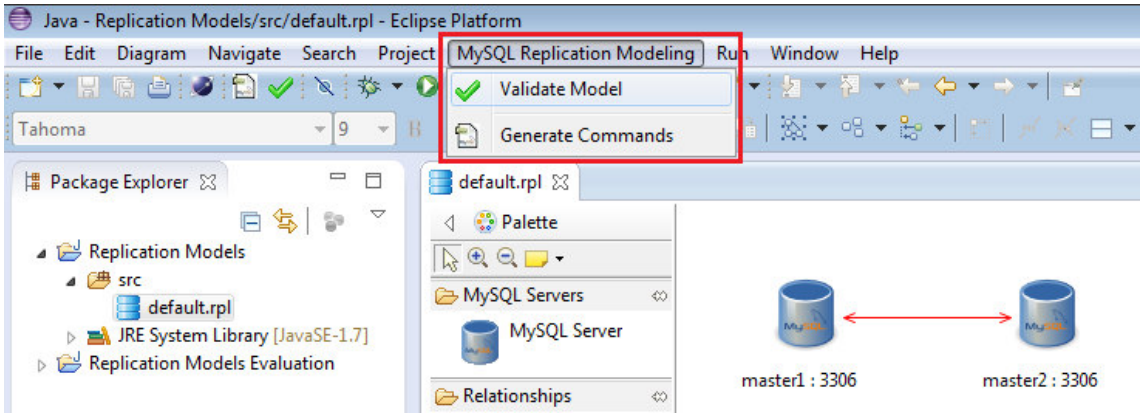


Luego de crear la relación maestro-maestro debe lucir como la imagen siguiente.

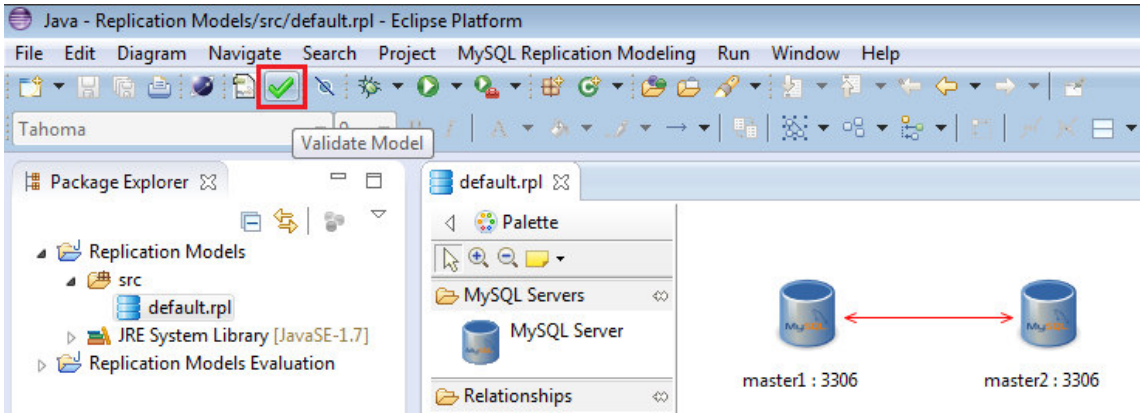


- **Validar Modelo de Replicación de MySQL**

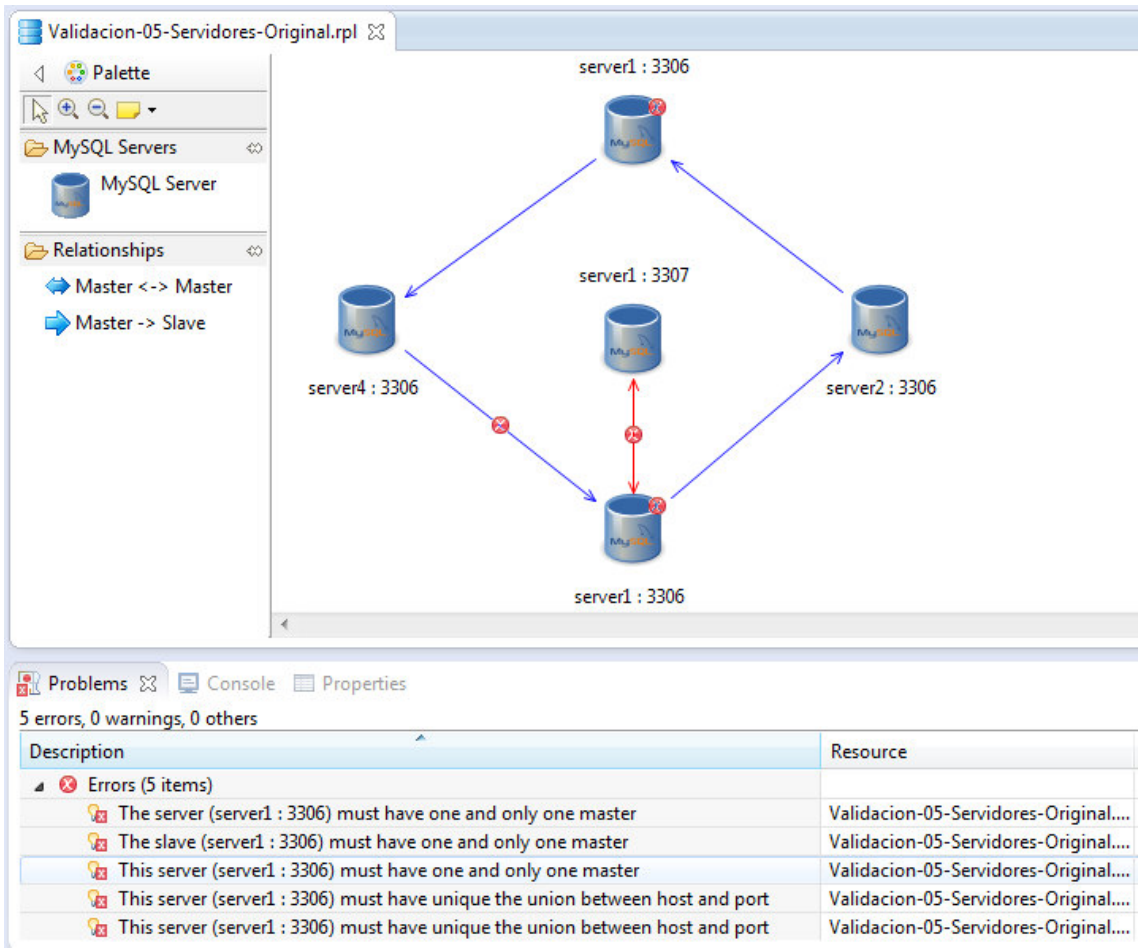
Para validar un modelo de replicación de MySQL, se debe seleccionar el menú **MySQL Replication Modeling** y dar click en la opción **Validate Model**.



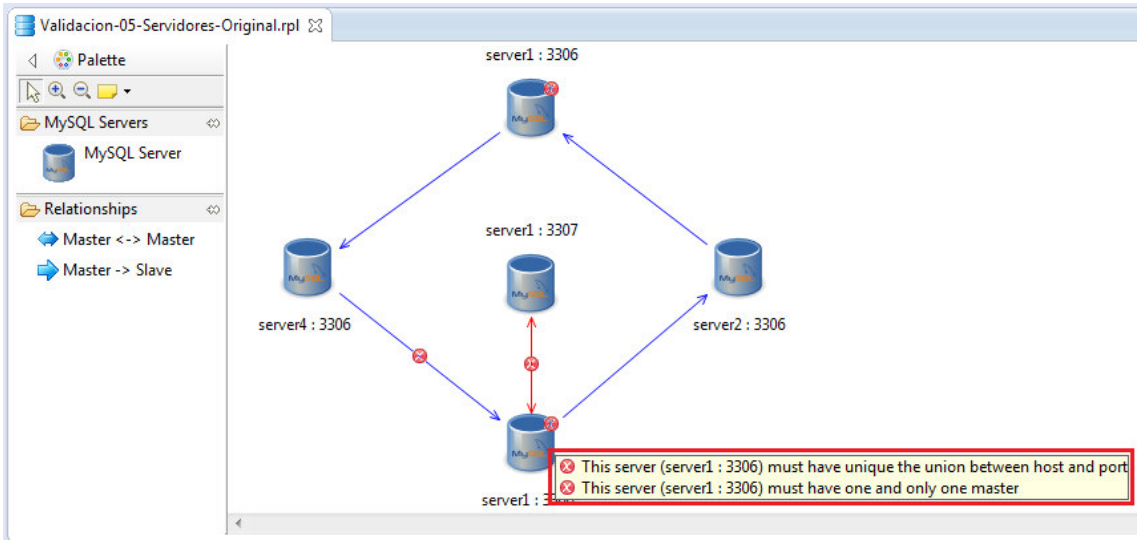
También se puede ejecutar la validación del modelo de replicación haciendo click en el icono de acceso directo, tal como se muestra en la siguiente figura.



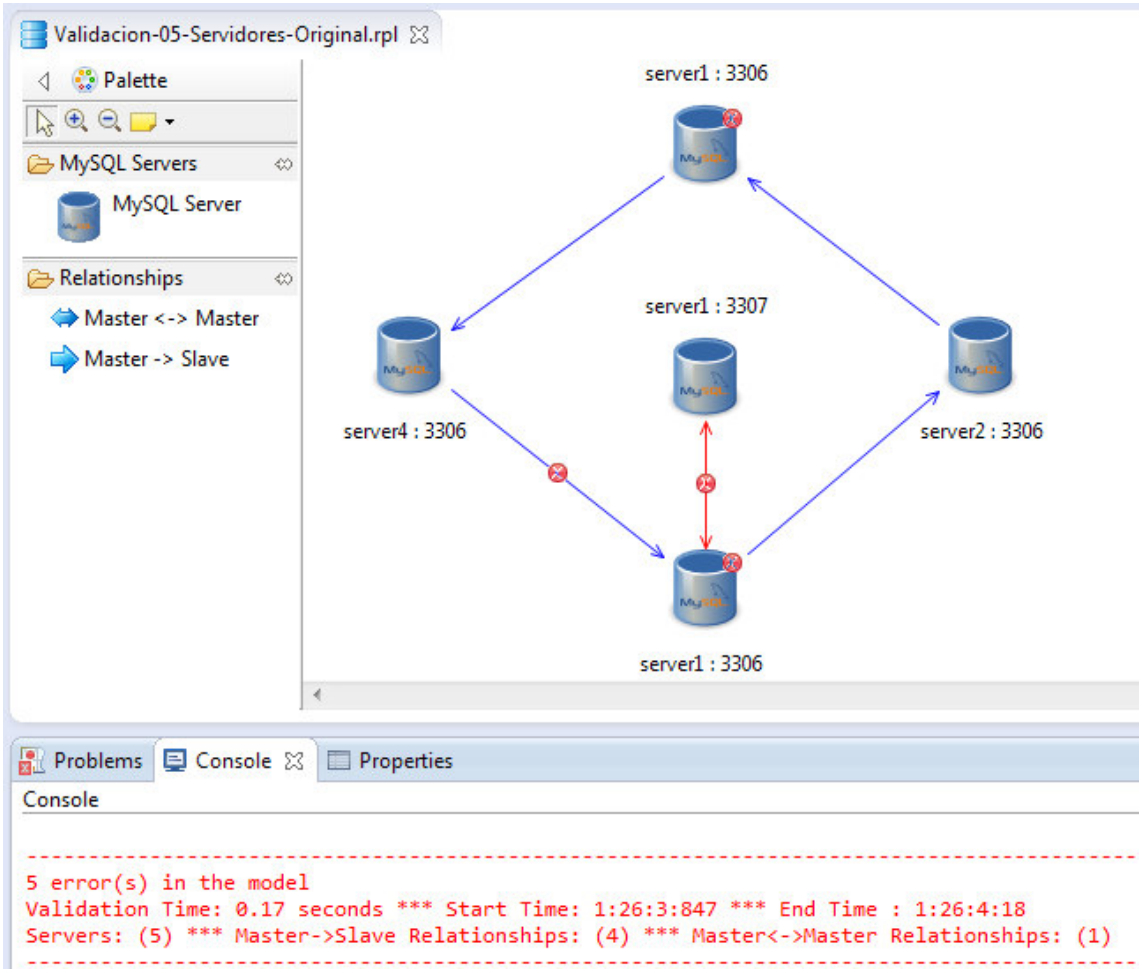
Luego de dar click en la opción **Validate Model** aparecerá una pantalla similar a la de la siguiente figura siempre y cuando un modelo contenga errores. Se puede notar en la figura que los errores se listan en la vista de problemas, y los servidores y relaciones con errores se marcan con un icono rojo.



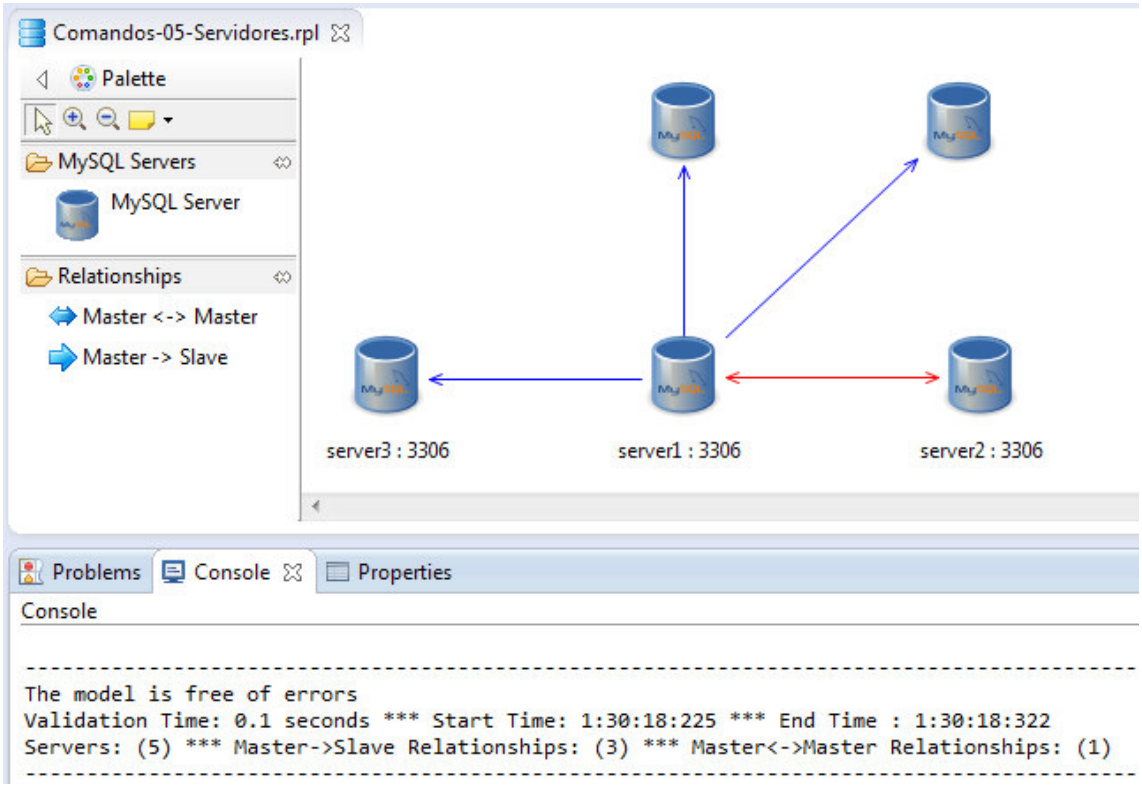
Al colocar el mouse sobre un icono rojo de error, aparecerá el mensaje indicando el error, tal como se muestra en la siguiente figura.



La herramienta también muestra en la vista de consola, el tiempo tomado en la validación, así como el número de servidores del modelo y el número de relaciones maestro-esclavo y maestro-maestro.

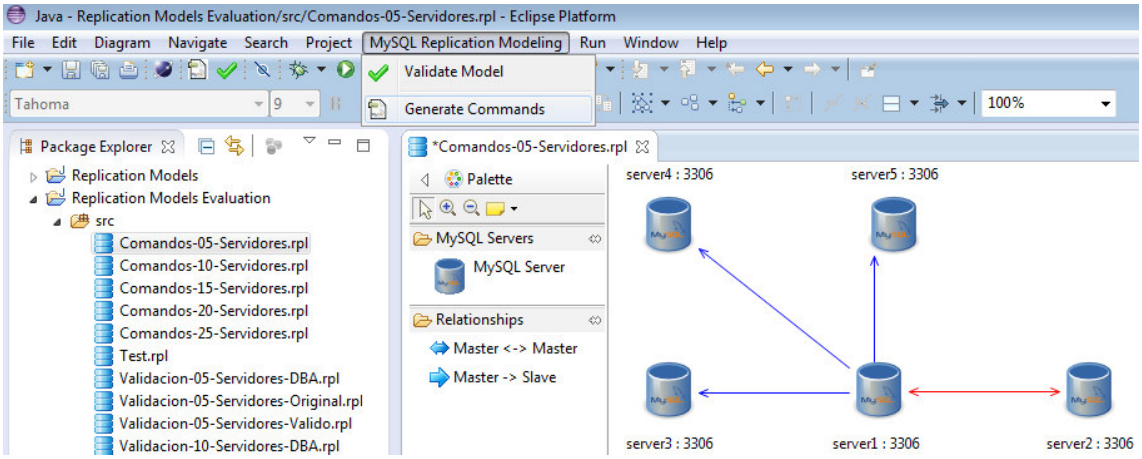


En caso un modelo de replicación no contenga errores, se mostrará una pantalla similar al de la siguiente figura.

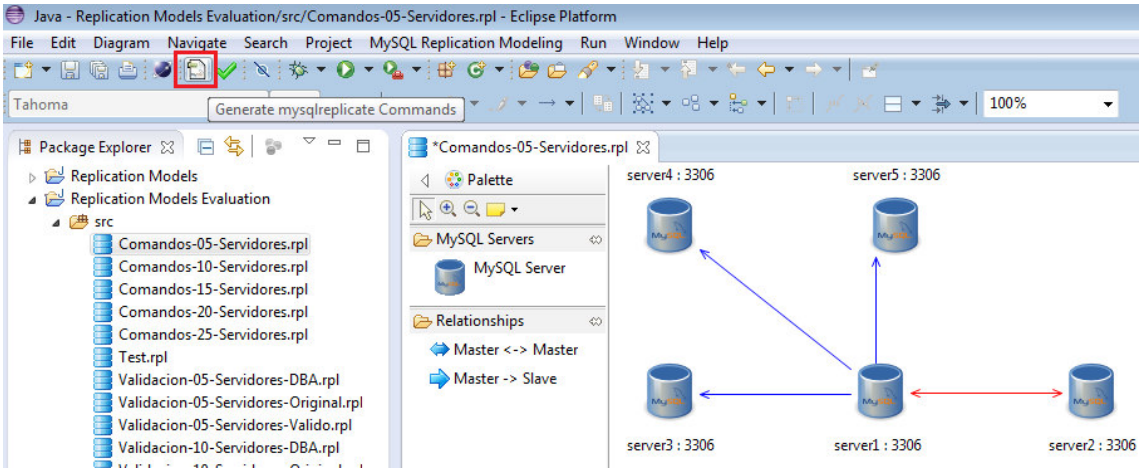


- **Generar Comandos de Configuración a partir del Modelo de Replicación**

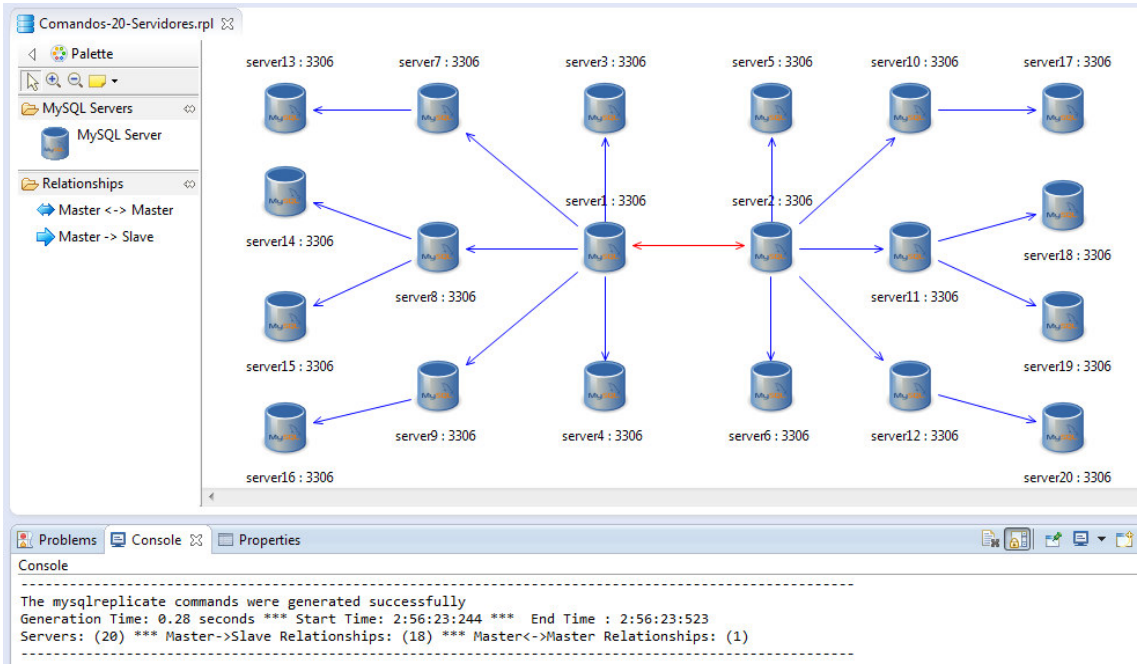
Para generar los comandos mysqlreplicate desde un modelo de replicación de MySQL, se debe seleccionar el menú **MySQL Replication Modeling** y dar click en la opción **Generate Commands,** tal como se muestra en la siguiente figura.



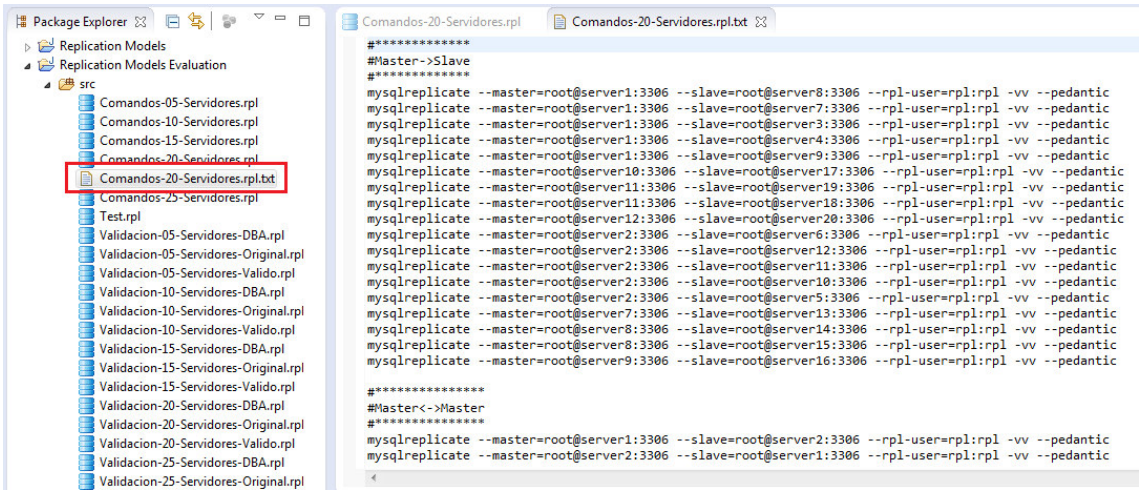
También se puede ejecutar la generación de comandos dando click en el icono de acceso directo, tal como se muestra en la siguiente figura.



Luego de dar click en la opción **Generate Commands**, la herramienta validará si el modelo de replicación es correcto para poder generar los comandos y mostrar un mensaje indicando que los comandos fueron generados satisfactoriamente, así como el tiempo tomado por el proceso de generación, tal como se muestra en la siguiente figura.



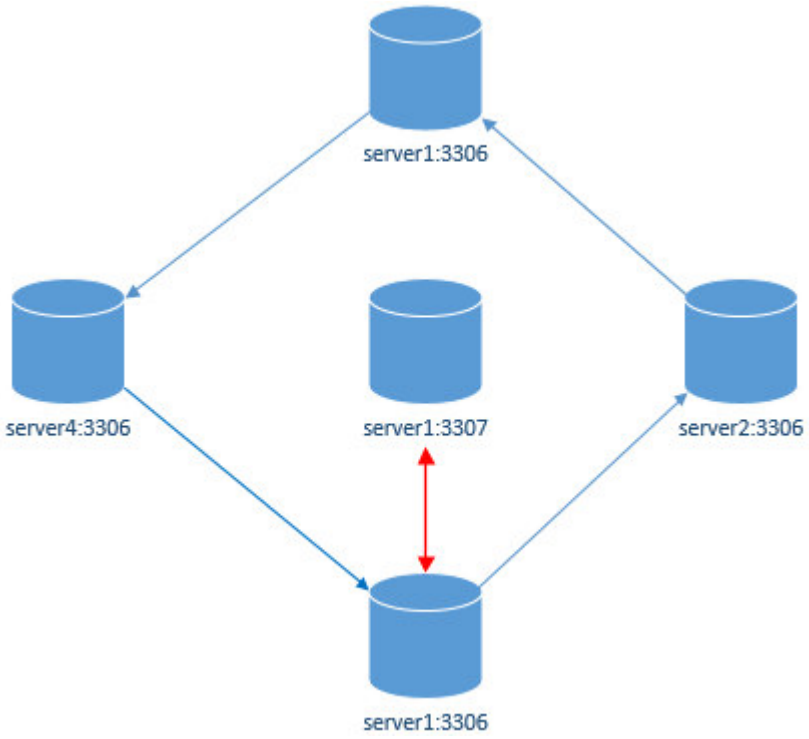
La herramienta generará un archivo de texto que contiene los comandos mysqlreplicate de configuración del modelo de replicación de MySQL, tal como se muestra en la siguiente figura.


Package Explorer
Replication Models
Replication Models Evaluation
src
Comandos-05-Servidores.rpl
Comandos-10-Servidores.rpl
Comandos-15-Servidores.rpl
Comandos-20-Servidores.rpl
Comandos-20-Servidores.rpl.txt
Comandos-25-Servidores.rpl
Test.rpl
Validacion-05-Servidores-DBA.rpl
Validacion-05-Servidores-Original.rpl
Validacion-05-Servidores-Valido.rpl
Validacion-10-Servidores-DBA.rpl
Validacion-10-Servidores-Original.rpl
Validacion-10-Servidores-Valido.rpl
Validacion-15-Servidores-DBA.rpl
Validacion-15-Servidores-Original.rpl
Validacion-15-Servidores-Valido.rpl
Validacion-20-Servidores-DBA.rpl
Validacion-20-Servidores-Original.rpl
Validacion-20-Servidores-Valido.rpl
Validacion-25-Servidores-DBA.rpl
Validacion-25-Servidores-Original.rpl
Comandos-20-Servidores.rpl
Comandos-20-Servidores.rpl.txt
#************
#Master->Slave
#************
mysqlreplicate --master=root@server1:3306 --slave=root@server8:3306 --rpl-user=rpl:rpl -vv --pedantic
mysqlreplicate --master=root@server1:3306 --slave=root@server7:3306 --rpl-user=rpl:rpl -vv --pedantic
mysqlreplicate --master=root@server1:3306 --slave=root@server3:3306 --rpl-user=rpl:rpl -vv --pedantic
mysqlreplicate --master=root@server1:3306 --slave=root@server4:3306 --rpl-user=rpl:rpl -vv --pedantic
mysqlreplicate --master=root@server1:3306 --slave=root@server9:3306 --rpl-user=rpl:rpl -vv --pedantic
mysqlreplicate --master=root@server10:3306 --slave=root@server17:3306 --rpl-user=rpl:rpl -vv --pedantic
mysqlreplicate --master=root@server11:3306 --slave=root@server19:3306 --rpl-user=rpl:rpl -vv --pedantic
mysqlreplicate --master=root@server11:3306 --slave=root@server18:3306 --rpl-user=rpl:rpl -vv --pedantic
mysqlreplicate --master=root@server12:3306 --slave=root@server20:3306 --rpl-user=rpl:rpl -vv --pedantic
mysqlreplicate --master=root@server2:3306 --slave=root@server6:3306 --rpl-user=rpl:rpl -vv --pedantic
mysqlreplicate --master=root@server2:3306 --slave=root@server12:3306 --rpl-user=rpl:rpl -vv --pedantic
mysqlreplicate --master=root@server2:3306 --slave=root@server11:3306 --rpl-user=rpl:rpl -vv --pedantic
mysqlreplicate --master=root@server2:3306 --slave=root@server10:3306 --rpl-user=rpl:rpl -vv --pedantic
mysqlreplicate --master=root@server2:3306 --slave=root@server5:3306 --rpl-user=rpl:rpl -vv --pedantic
mysqlreplicate --master=root@server7:3306 --slave=root@server13:3306 --rpl-user=rpl:rpl -vv --pedantic
mysqlreplicate --master=root@server8:3306 --slave=root@server14:3306 --rpl-user=rpl:rpl -vv --pedantic
mysqlreplicate --master=root@server8:3306 --slave=root@server15:3306 --rpl-user=rpl:rpl -vv --pedantic
mysqlreplicate --master=root@server9:3306 --slave=root@server16:3306 --rpl-user=rpl:rpl -vv --pedantic
#***************
#Master<->Master
#***************
mysqlreplicate --master=root@server1:3306 --slave=root@server2:3306 --rpl-user=rpl:rpl -vv --pedantic
mysqlreplicate --master=root@server2:3306 --slave=root@server1:3306 --rpl-user=rpl:rpl -vv --pedantic

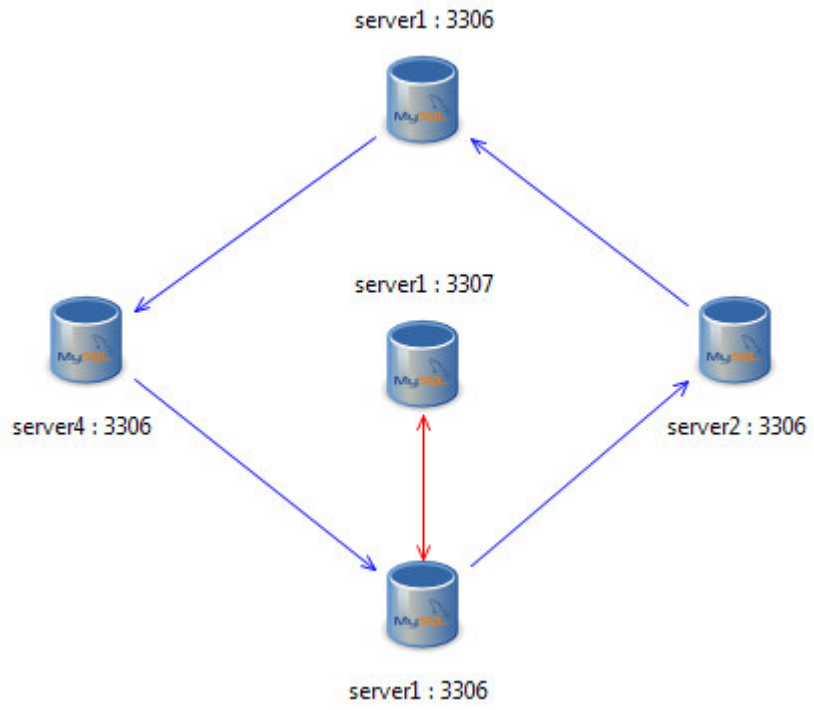
Anexo 4: Modelos de Replicación para la Corrección de Errores

Modelo de Replicación con 5 Servidores – Corrección del Modelo

Microsoft Visio 2013

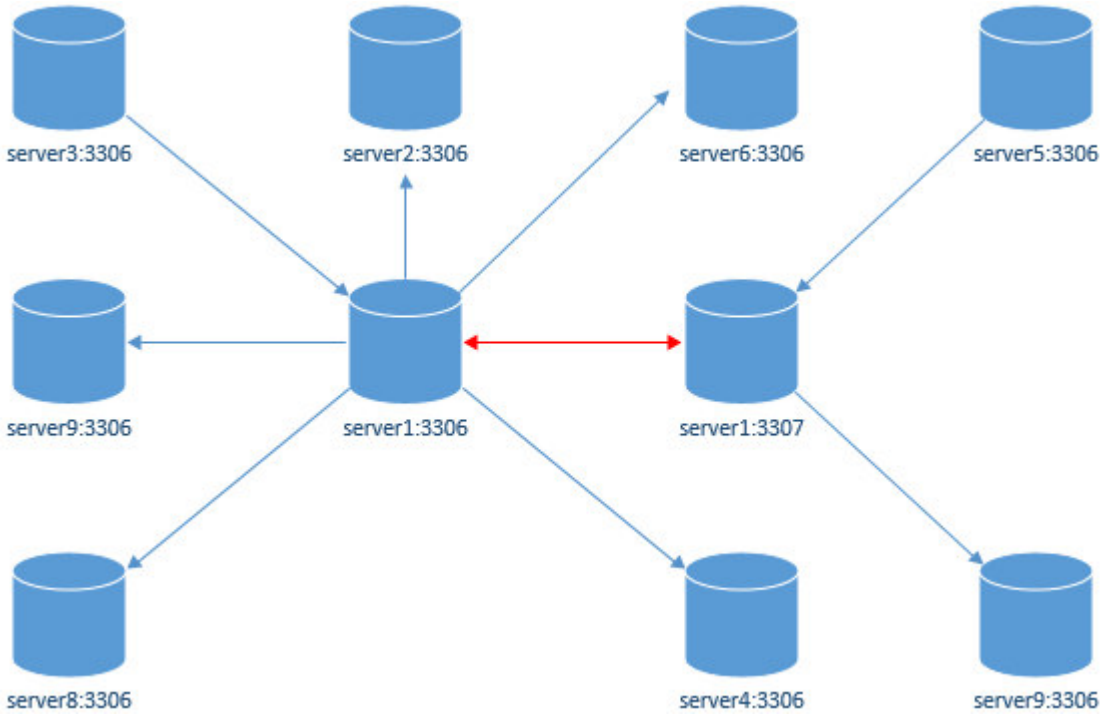


MySQL Replication Modeling

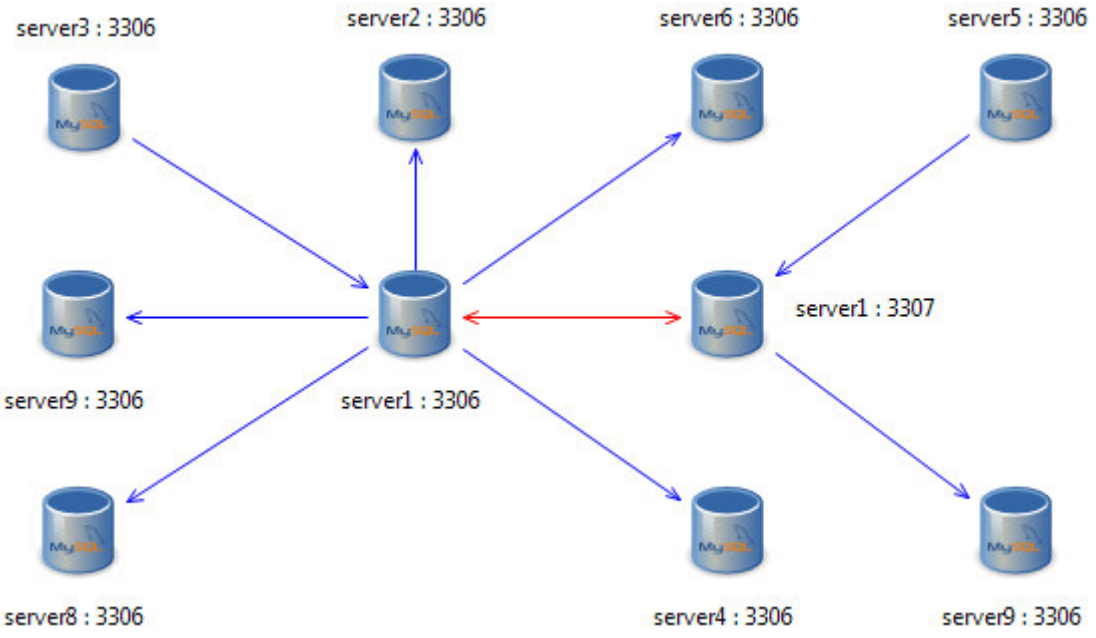


Modelo de Replicación con 10 Servidores – Corrección del Modelo

Microsoft Visio 2013

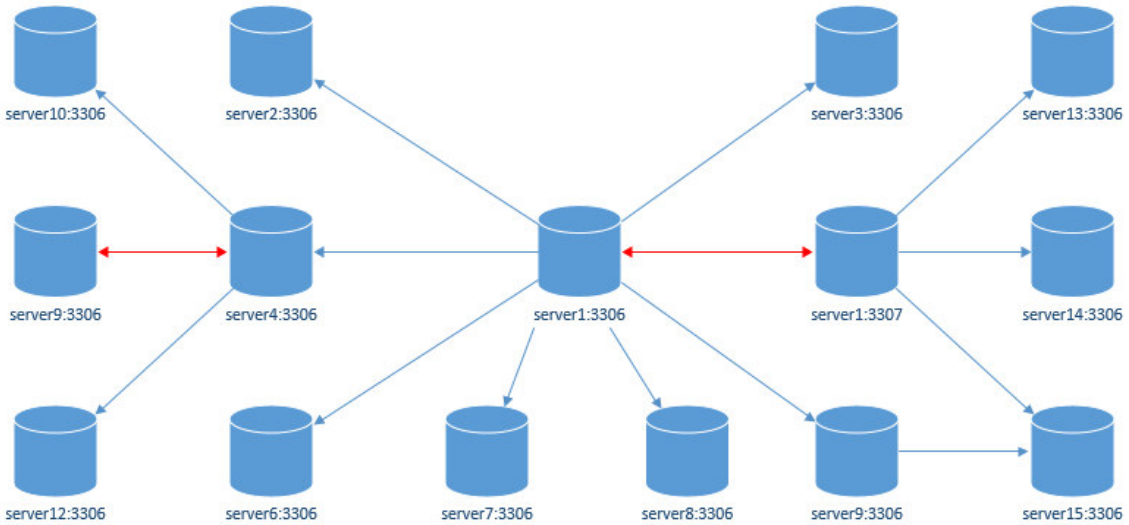


MySQL Replication Modeling

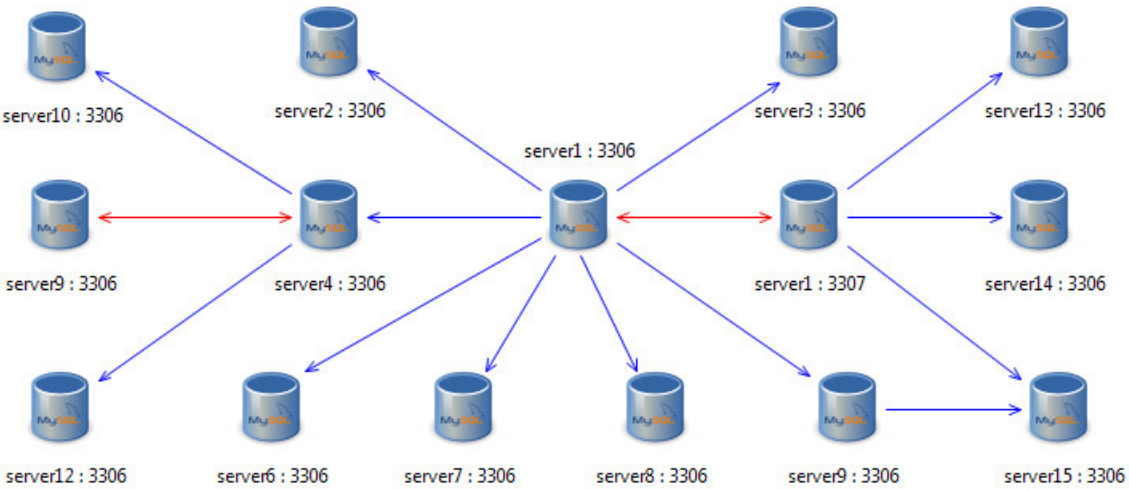


Modelo de Replicación con 15 Servidores – Corrección del Modelo

Microsoft Visio 2013

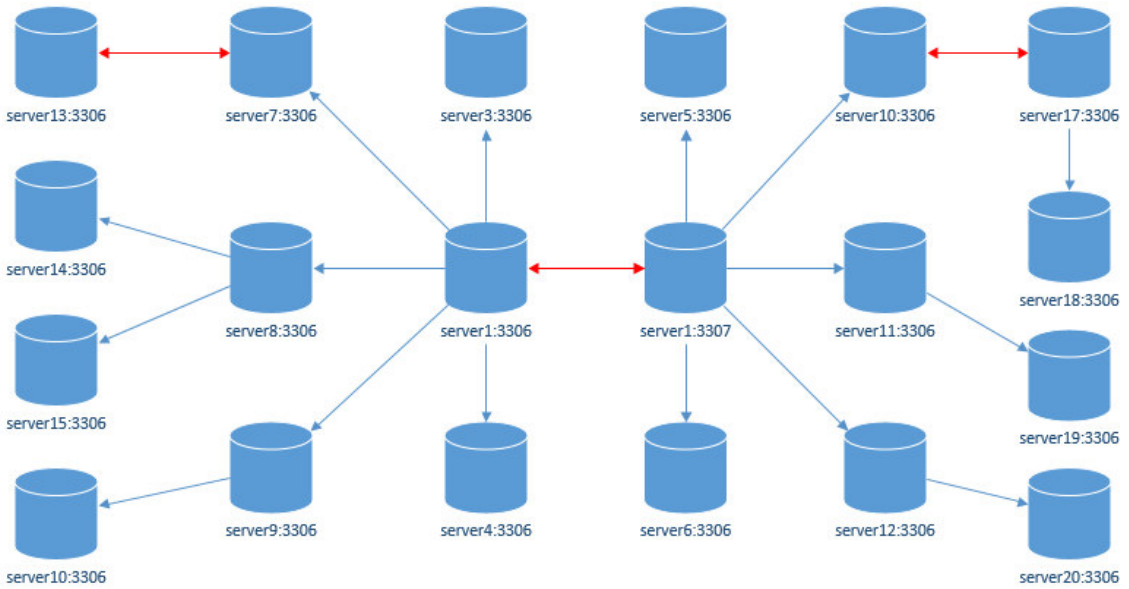


MySQL Replication Modeling

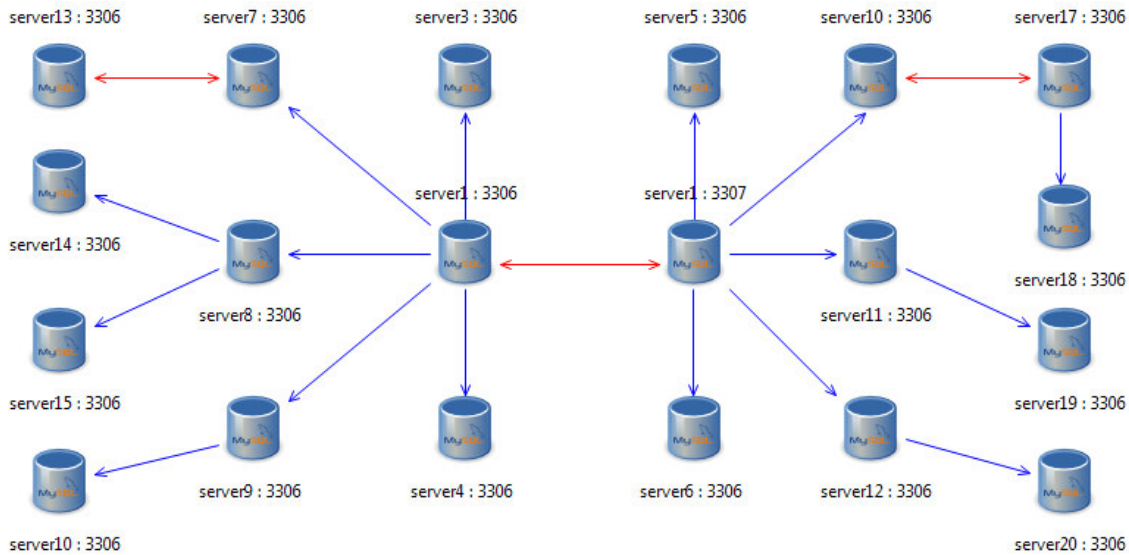


Modelo de Replicación con 20 Servidores – Corrección del Modelo

Microsoft Visio 2013

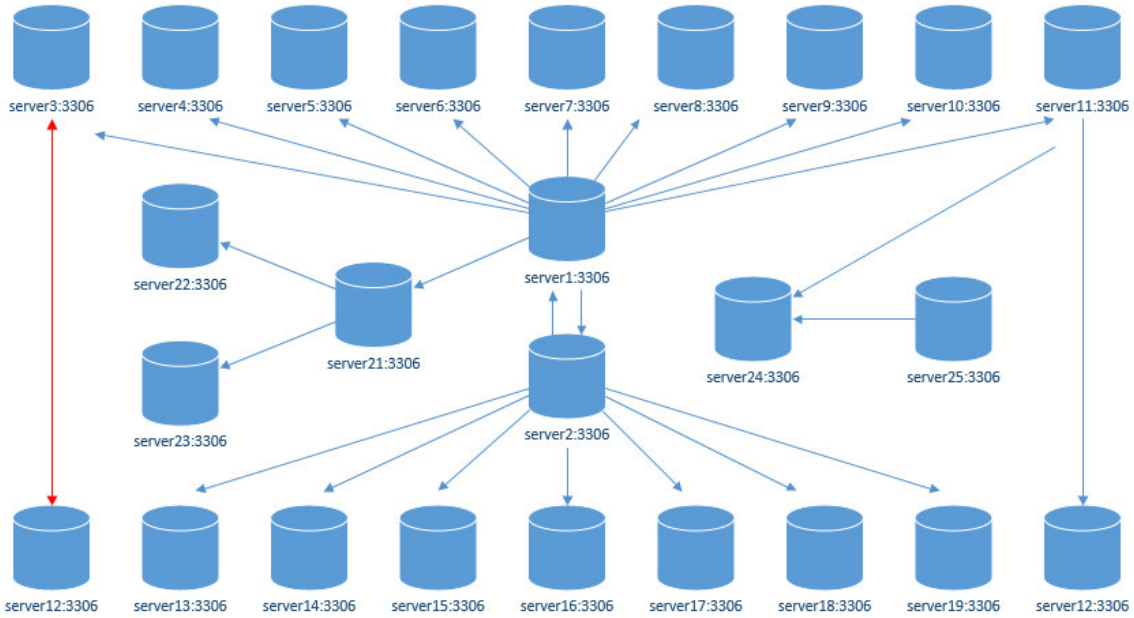


MySQL Replication Modeling

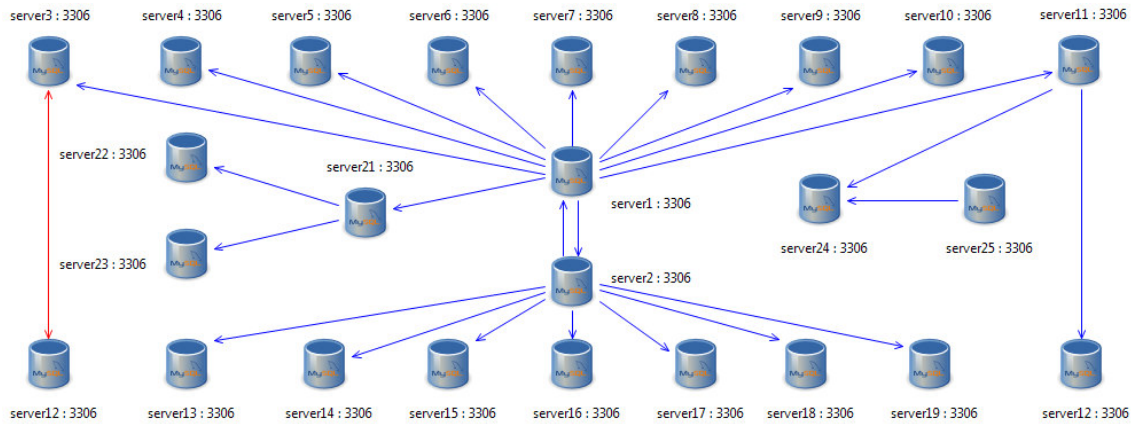


Modelo de Replicación con 25 Servidores – Corrección del Modelo

Microsoft Visio 2013



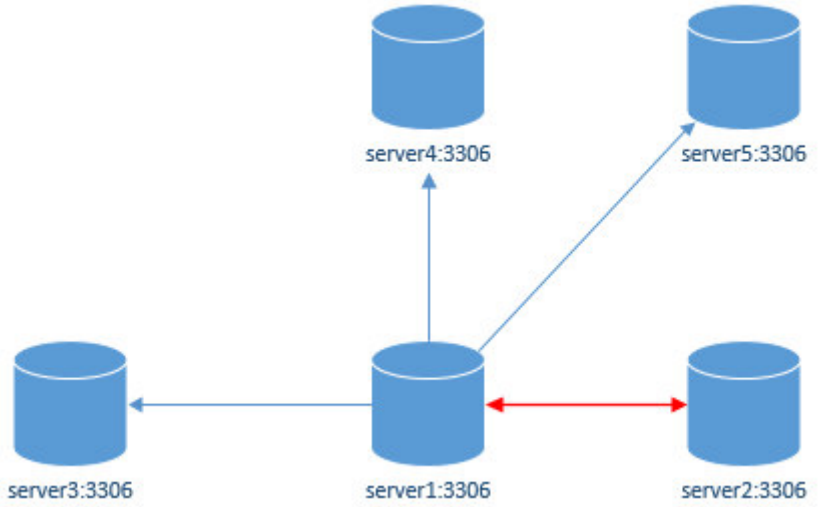
MySQL Replication Modeling



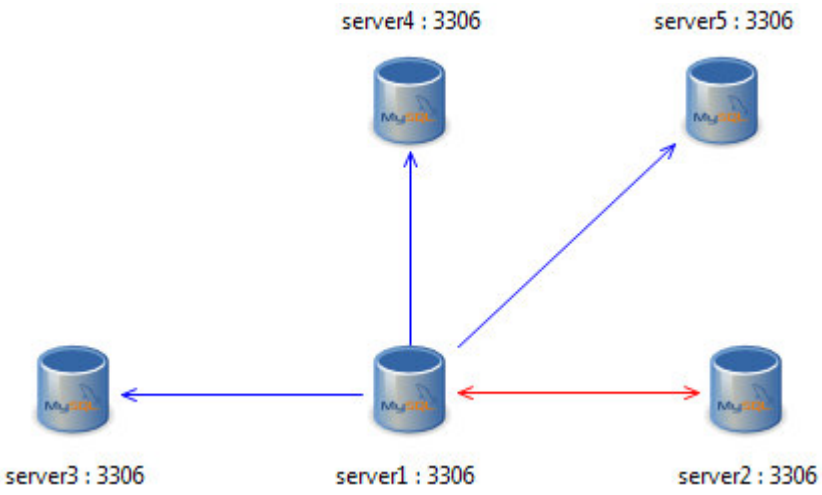
Anexo 5: Modelos de Replicación para la Generación de Comandos mysqlreplicate de Configuración

Modelo de Replicación con 5 Servidores – Generación de Comandos

Microsoft Visio 2013

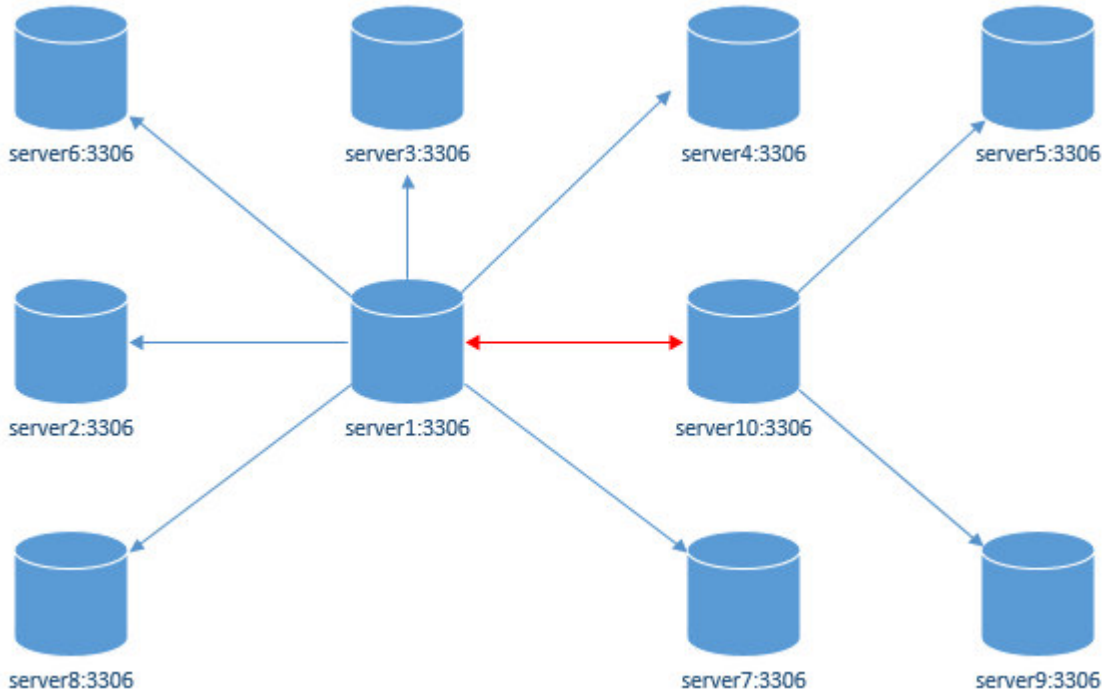


MySQL Replication Modeling

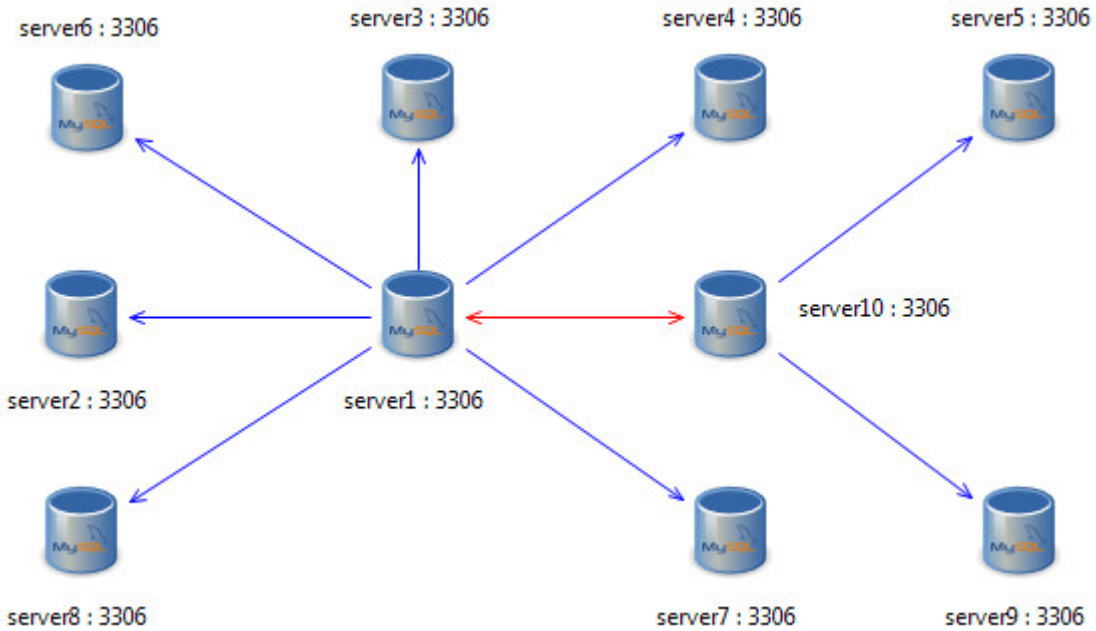


Modelo de Replicación con 10 Servidores – Generación de Comandos

Microsoft Visio 2013

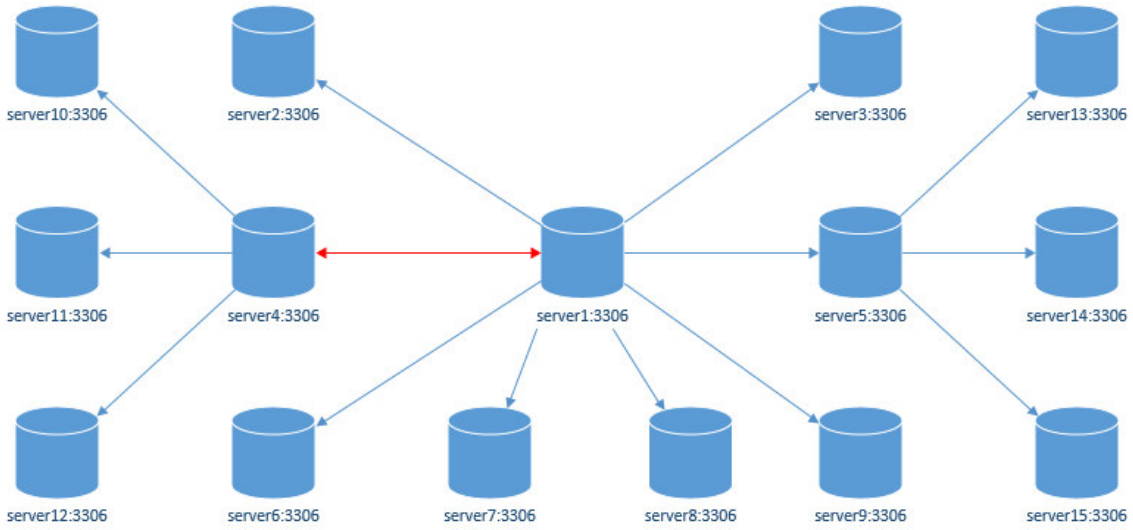


MySQL Replication Modeling

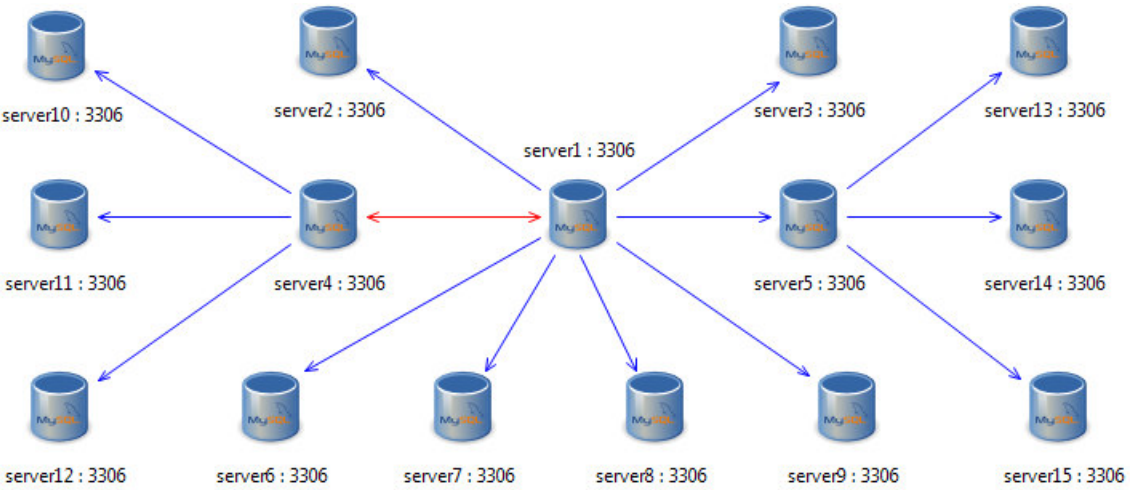


Modelo de Replicación con 15 Servidores – Generación de Comandos

Microsoft Visio 2013

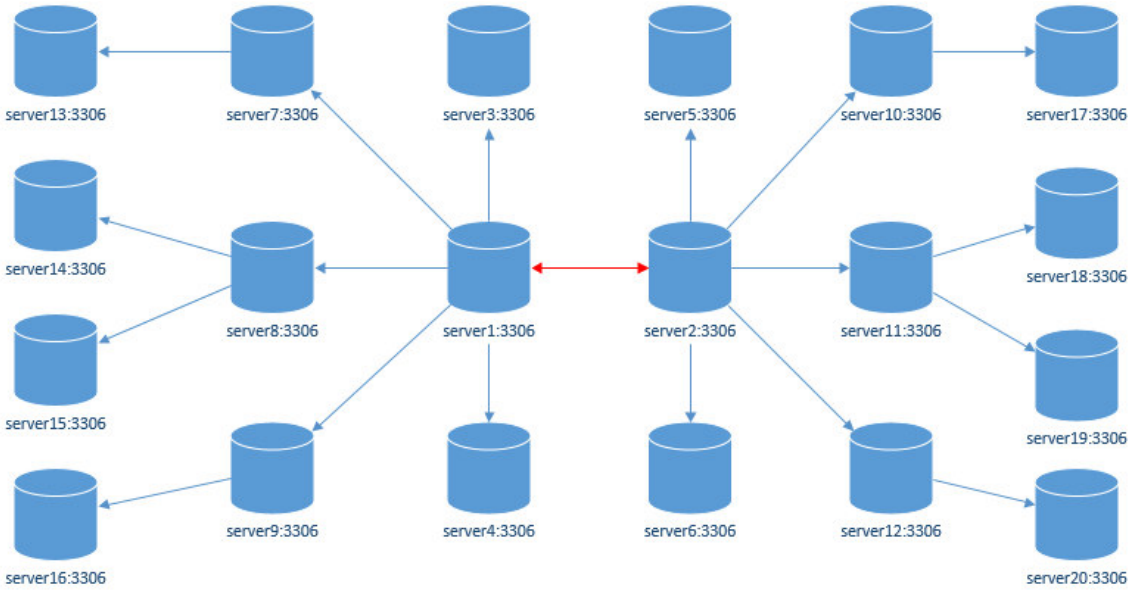


MySQL Replication Modeling

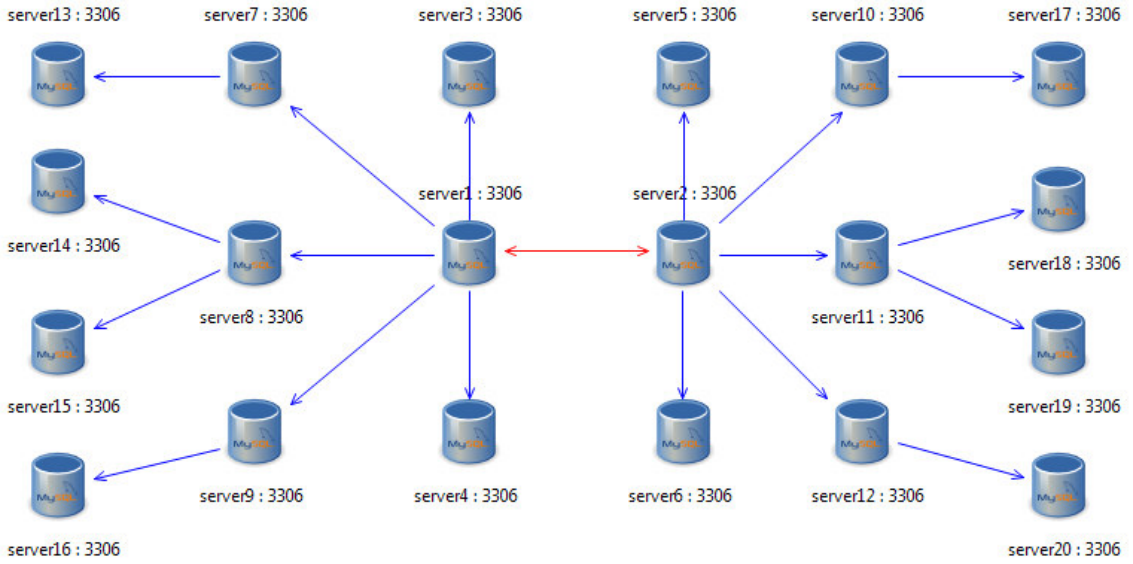


Modelo de Replicación con 20 Servidores – Generación de Comandos

Microsoft Visio 2013

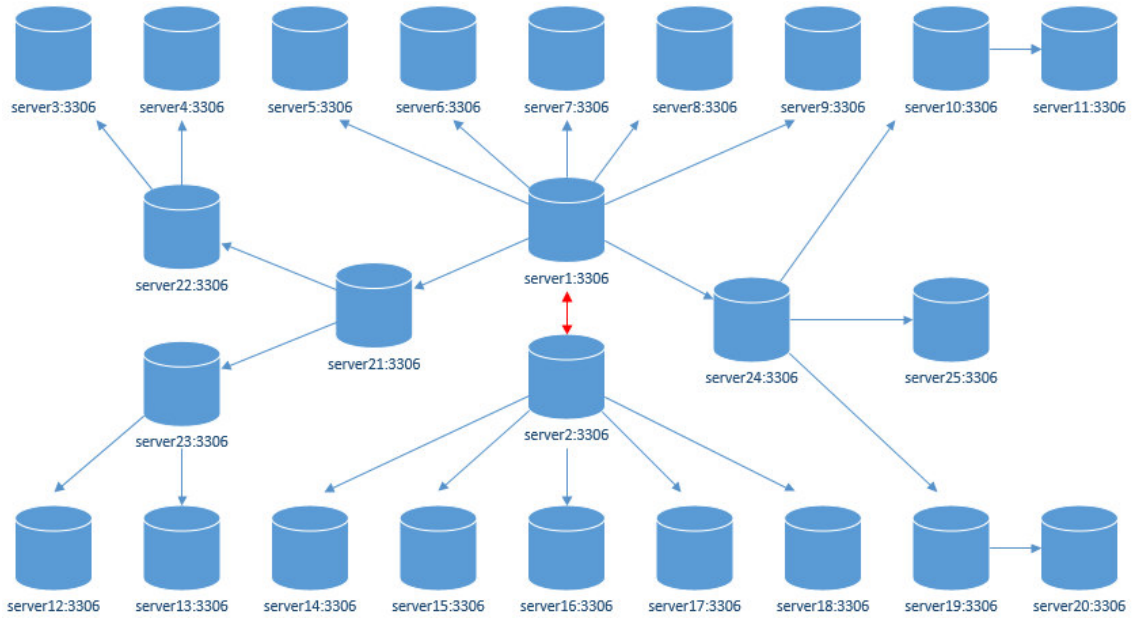


MySQL Replication Modeling

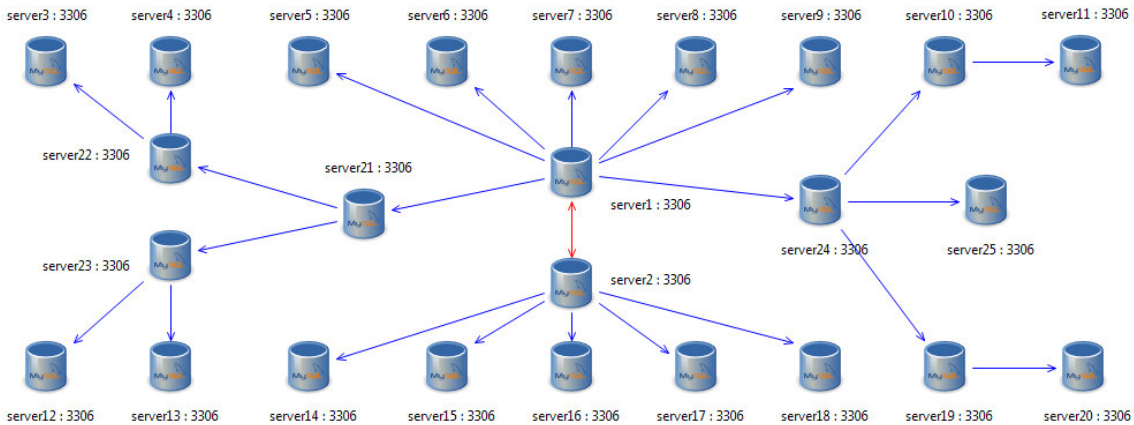


Modelo de Replicación con 25 Servidores – Generación de Comandos

Microsoft Visio 2013



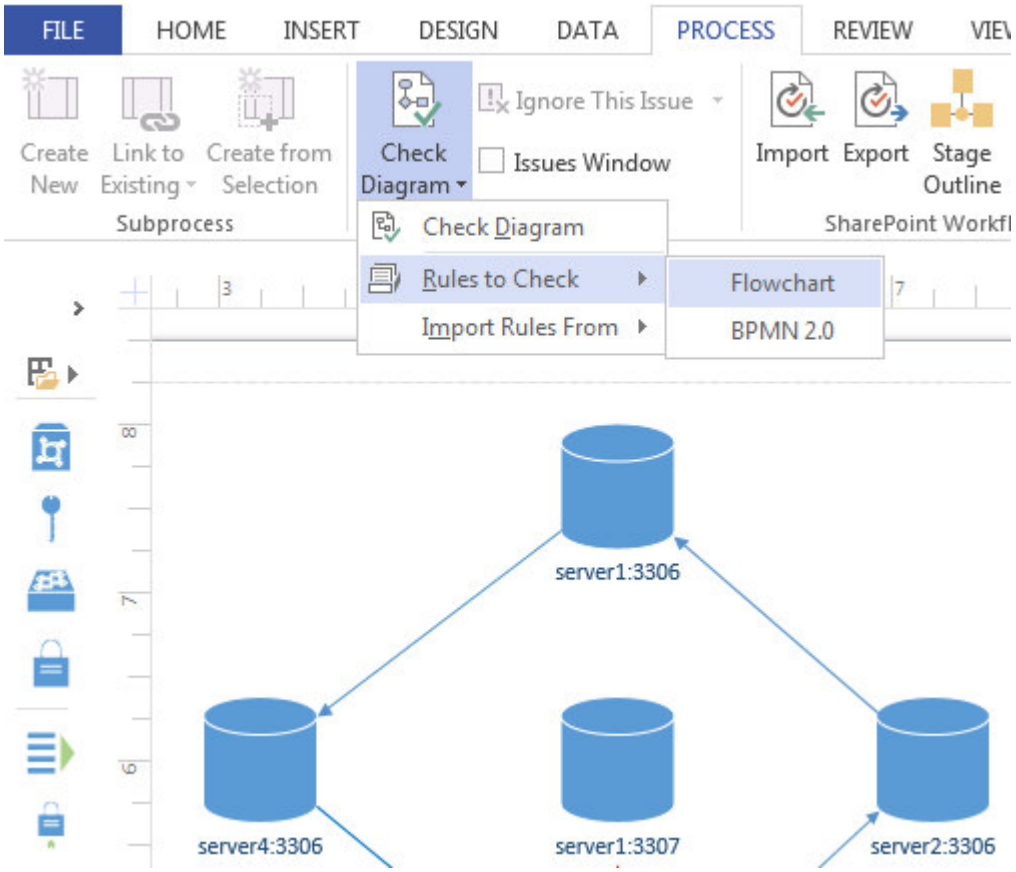
MySQL Replication Modeling



Anexo 6: Aplicación de Herramientas para la Corrección de Errores

La herramienta Microsoft Visio 2013 permite la opción de validar un diagrama de tipo Flowchart y BPMN 2.0, tal como se muestra en la siguiente figura.

Sin embargo, Microsoft Visio 2013 no permite a un DBA validar un modelo de replicación de MySQL, es por ello, que para corregir un modelo de replicación, un DBA tiene que identificar manualmente los errores.

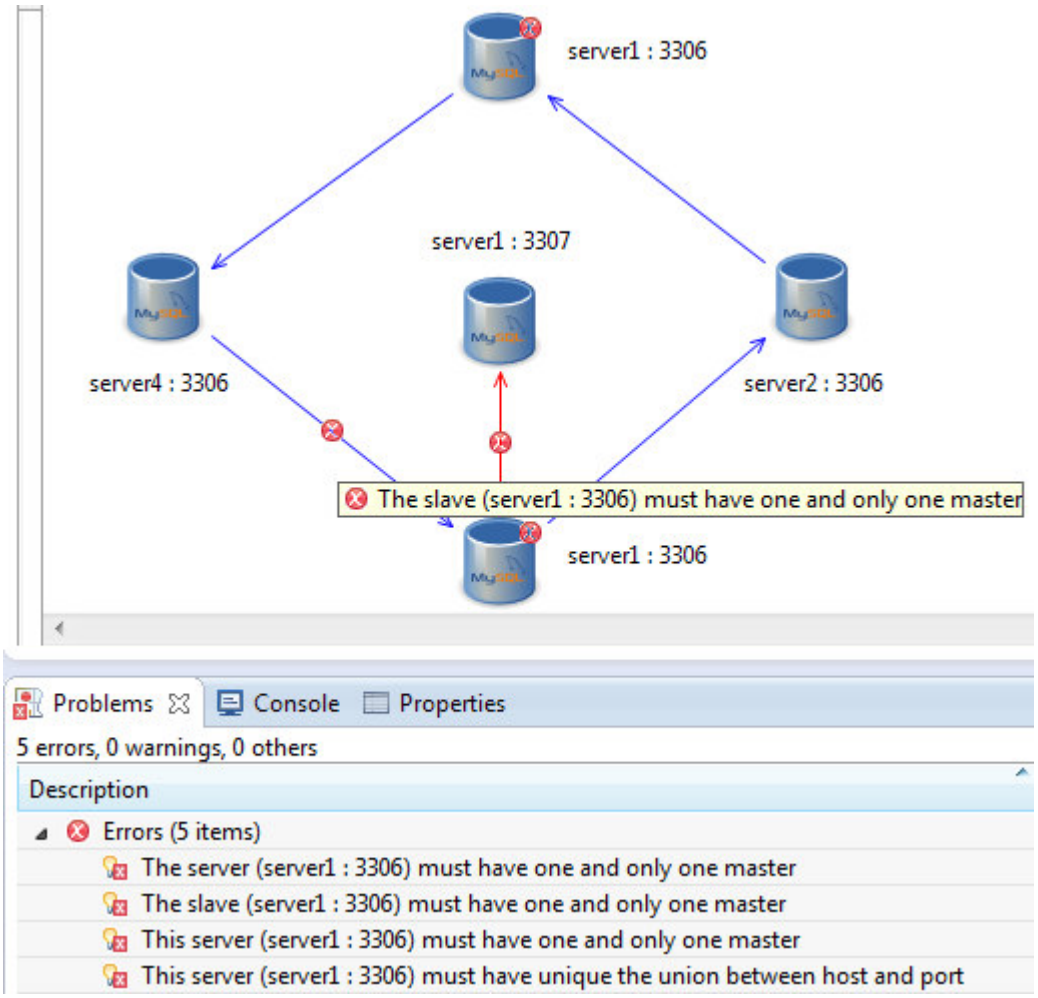


La herramienta propuesta MySQL Replication Modeling, sí permite validar un modelo de replicación de MySQL, en la que se listan los errores del modelo en una vista de problemas, y al dar click en cualquier error de esa vista, se selecciona según sea el caso el servidor o relación de replicación que presenta errores.

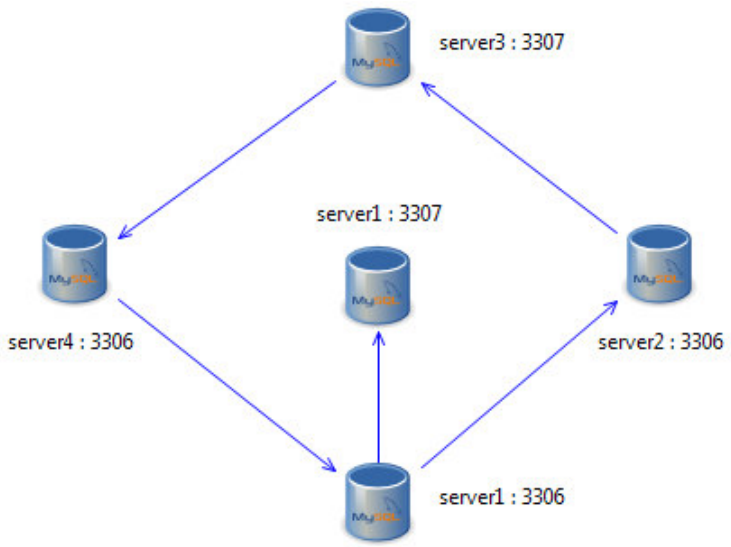
La herramienta propuesta también marca con un icono rojo los servidores y relaciones de replicación que tienen errores, todo esto para una rápida identificación de los errores de modelo por parte del DBA, de modo tal que pueda corregir el modelo de replicación MySQL en menos tiempo.

Uso de Herramientas – Corrección de un Modelo con 5 Servidores

En la siguiente figura se muestra la aplicación de MySQL Replication Modeling para la instancia de prueba de la validación de un modelo de replicación con 5 servidores.

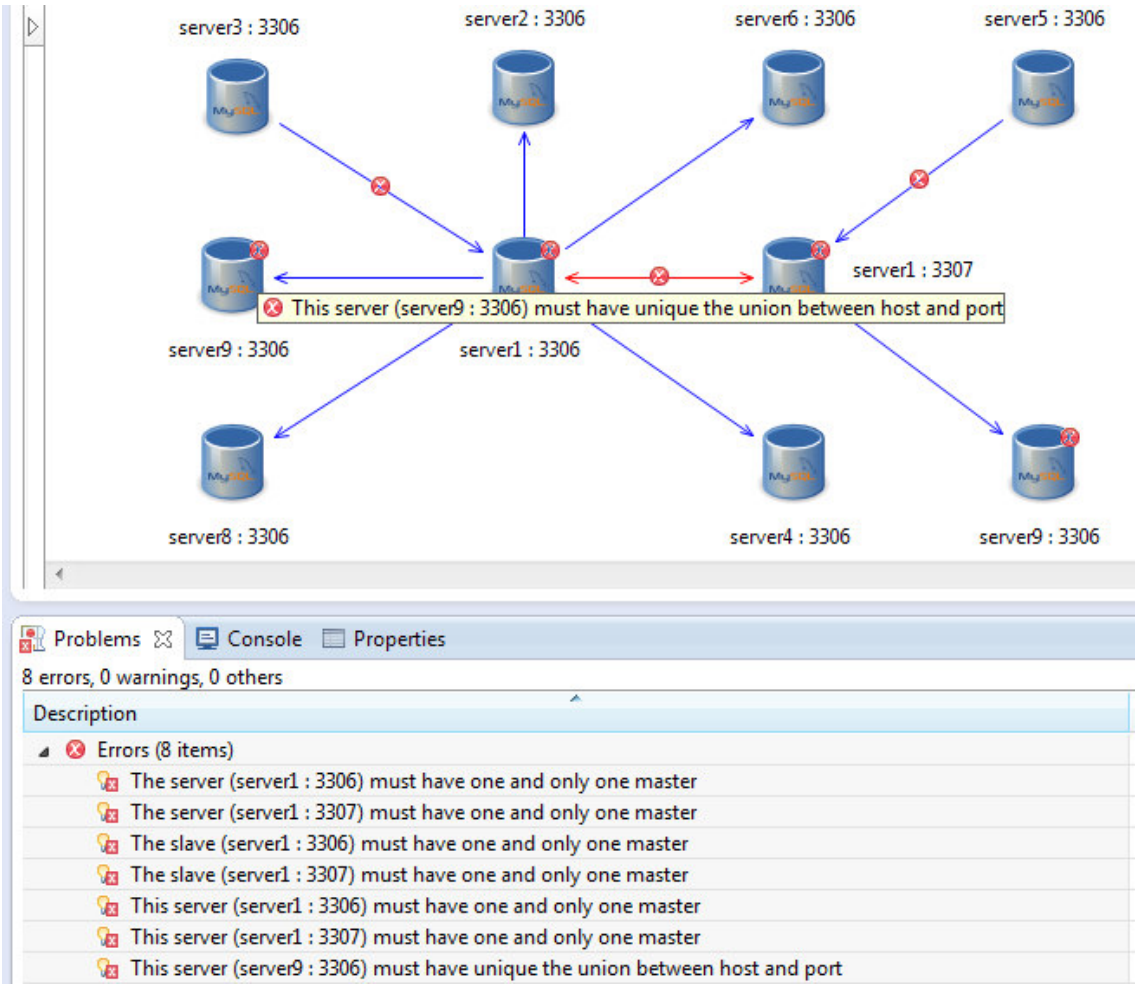


El modelo de replicación corregido por el DBA se muestra en la siguiente figura.

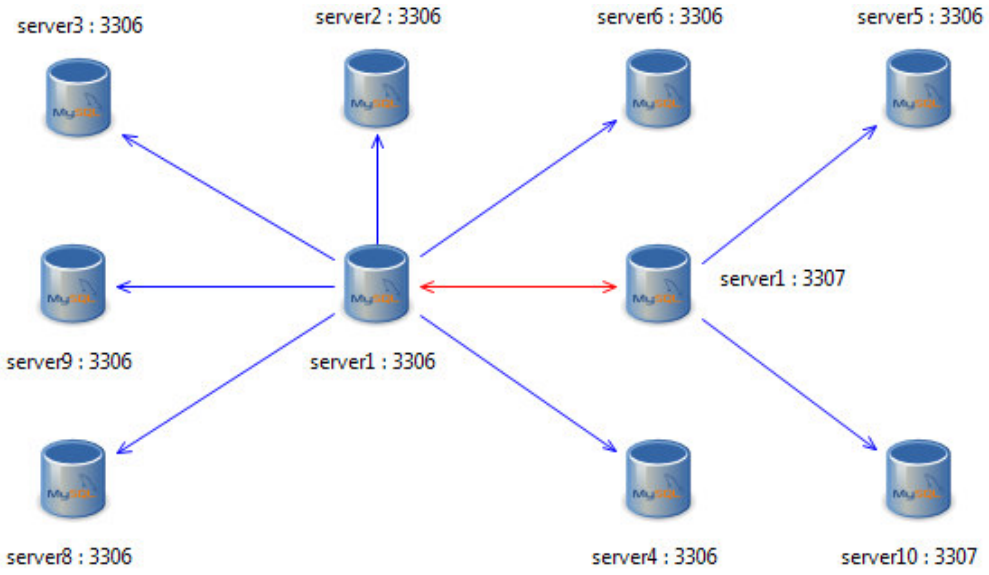


Uso de Herramientas – Corrección de un Modelo con 10 Servidores

En la siguiente figura se muestra la aplicación de MySQL Replication Modeling para la instancia de prueba de la validación de un modelo de replicación con 10 servidores.

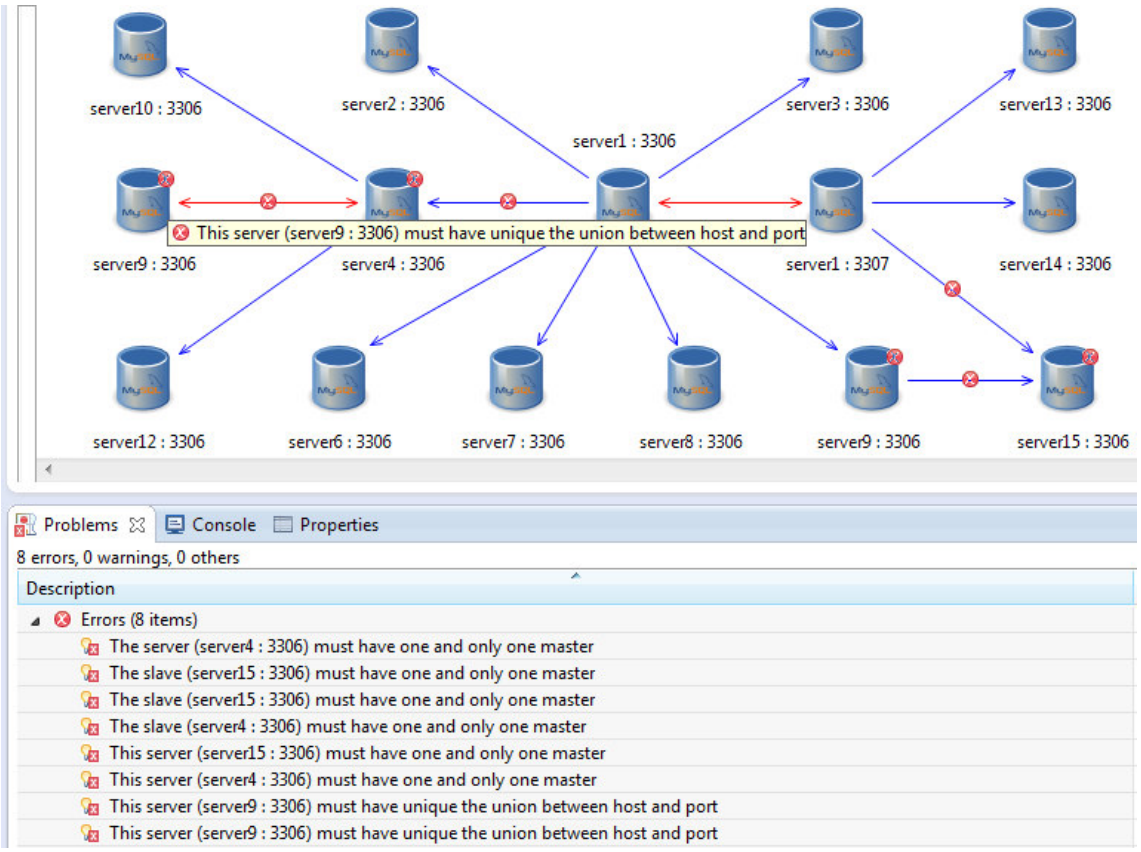


El modelo de replicación corregido por el DBA se muestra en la siguiente figura.

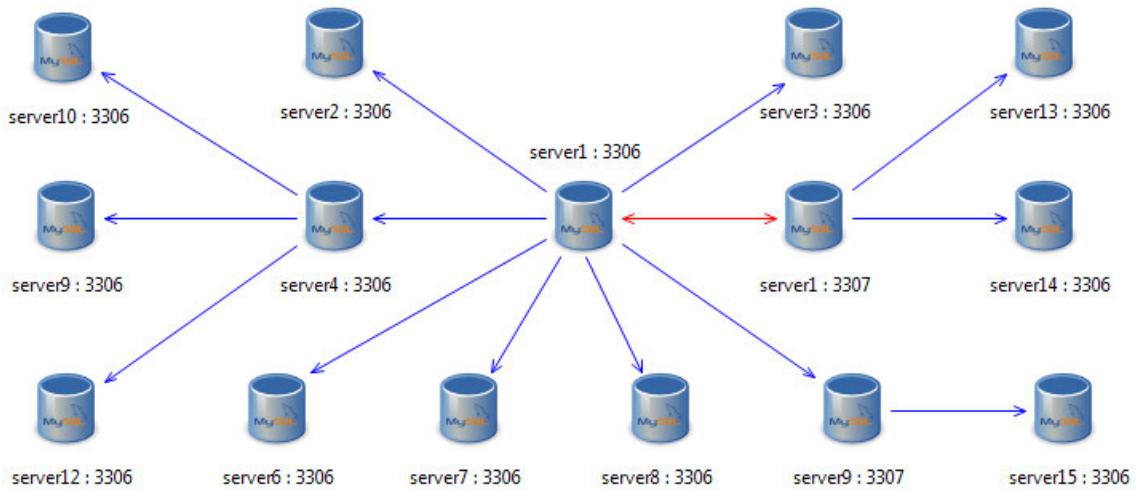


Uso de Herramientas – Corrección de un Modelo con 15 Servidores

En la siguiente figura se muestra la aplicación de MySQL Replication Modeling para la instancia de prueba de la validación de un modelo de replicación con 15 servidores.

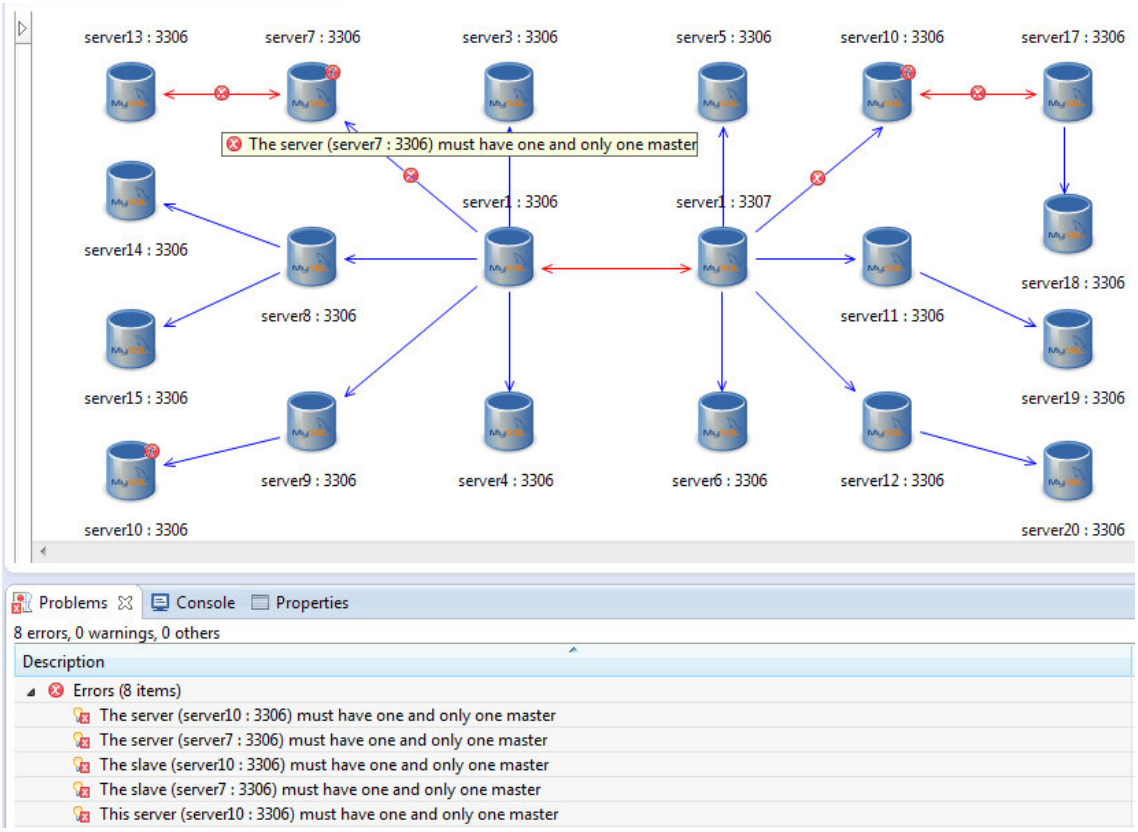


El modelo de replicación corregido por el DBA se muestra en la siguiente figura.

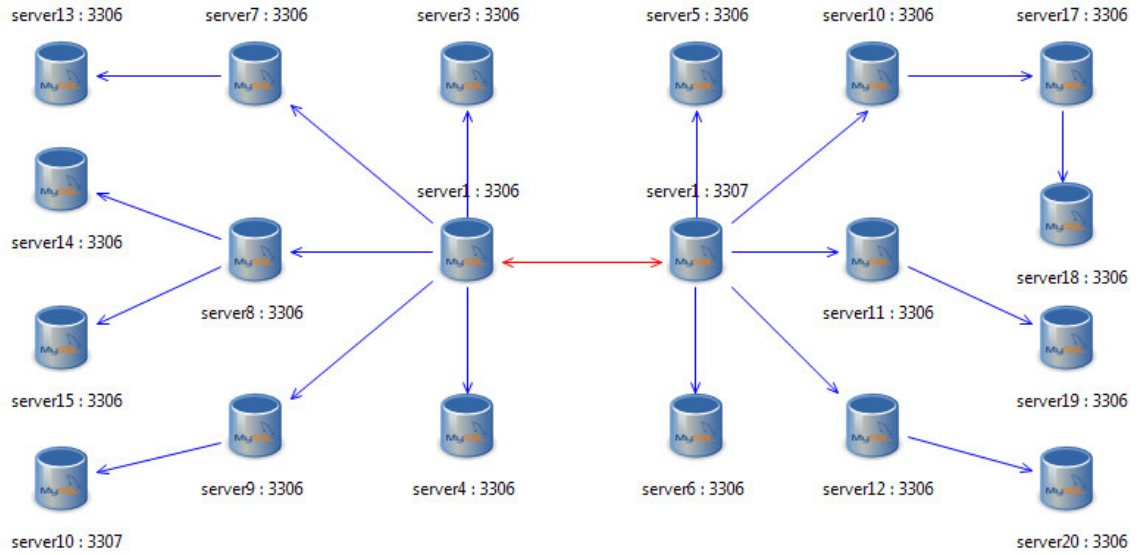


Uso de Herramientas – Corrección de un Modelo con 20 Servidores

En la siguiente figura se muestra la aplicación de MySQL Replication Modeling para la instancia de prueba de la validación de un modelo de replicación con 20 servidores.

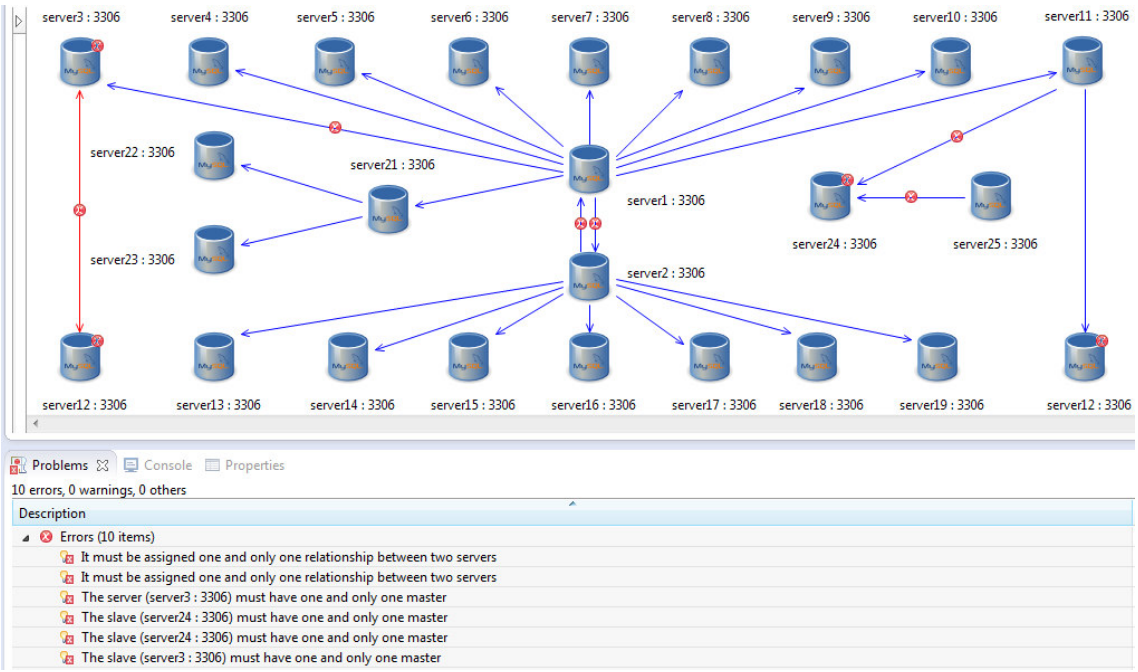


El modelo de replicación corregido por el DBA se muestra en la siguiente figura.

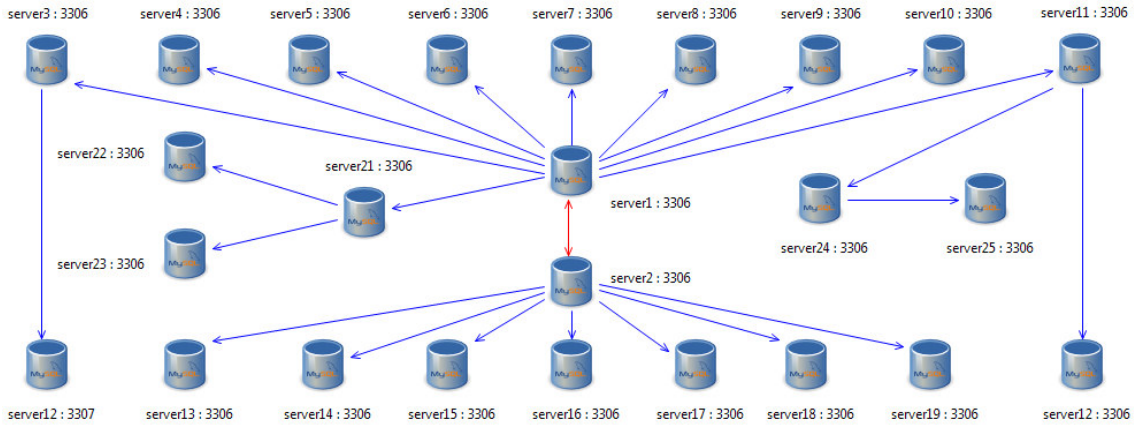


Uso de Herramientas – Corrección de un Modelo con 25 Servidores

En la siguiente figura se muestra la aplicación de MySQL Replication Modeling para la instancia de prueba de la validación de un modelo de replicación con 25 servidores.



El modelo de replicación corregido por el DBA se muestra en la siguiente figura.

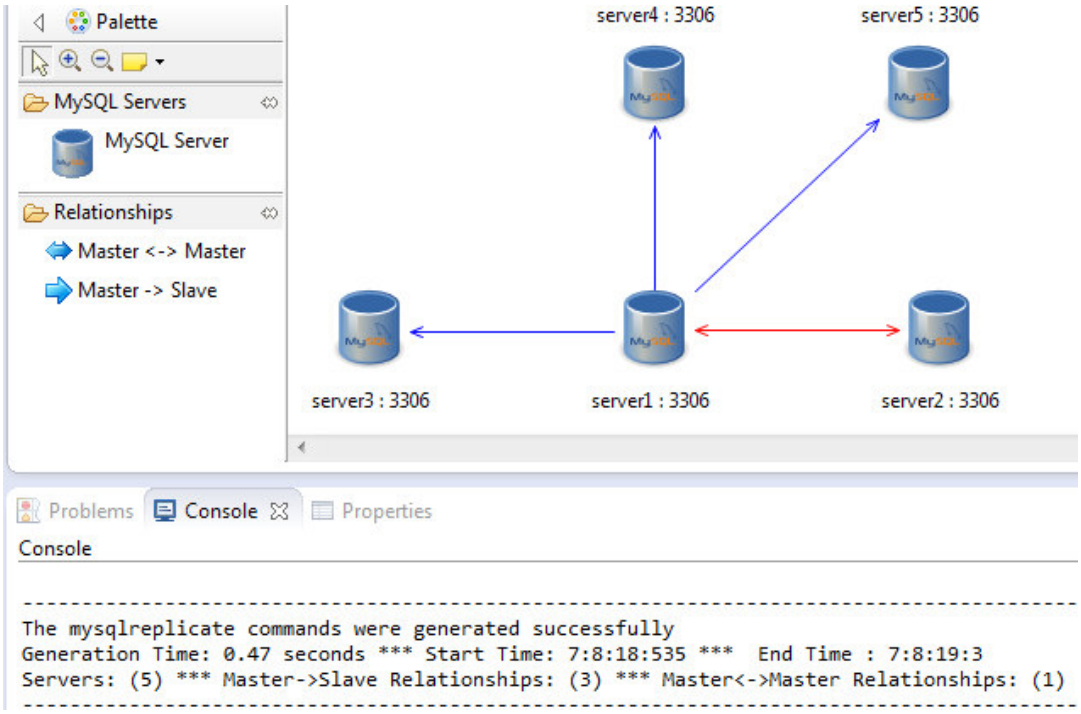


Anexo 7: Aplicación de Herramientas para la Generación de Comandos mysqlreplicate de Configuración

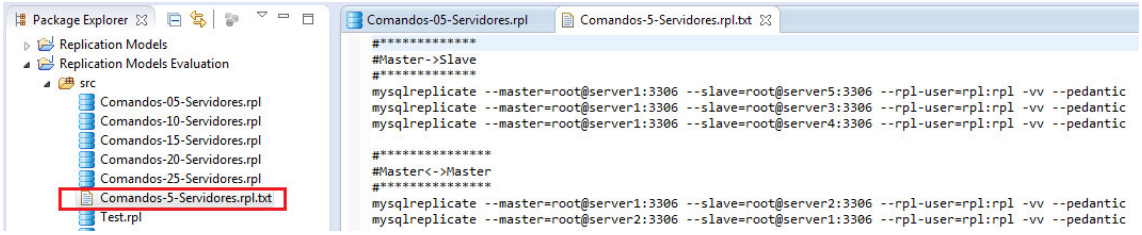
La herramienta Microsoft Visio 2013 no permite generar automáticamente los comandos de configuración a partir de un modelo de replicación de MySQL, por lo que un DBA los tienen que escribir manualmente. La herramienta propuesta MySQL Replication Modeling sí permite generar automáticamente los comandos de configuración a partir de un modelo de replicación de MySQL.

Uso de Herramientas – Generación de Comandos de un Modelo con 5 Servidores

En la siguiente figura se muestra la aplicación de la herramienta MySQL Replication Modeling para la instancia de prueba de generación de comandos a partir de un modelo de replicación con 10 servidores MySQL.

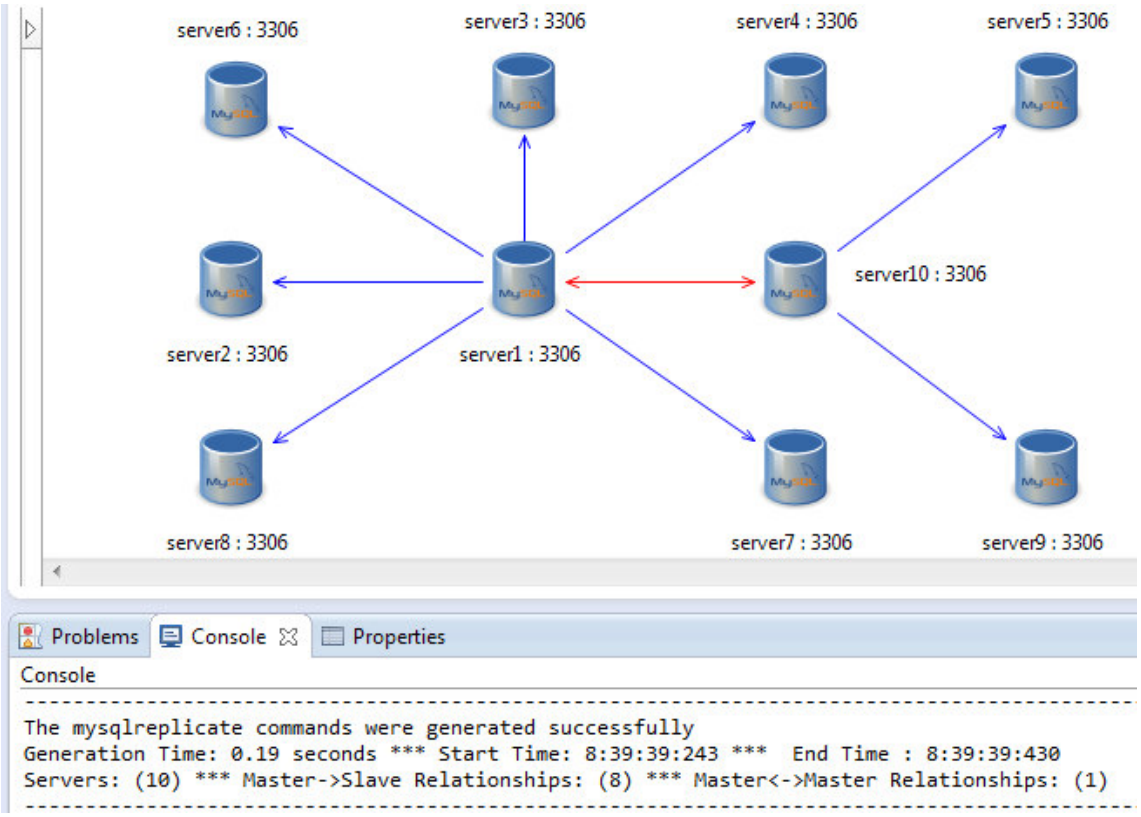


La herramienta generará un archivo de texto en el proyecto con los comandos para configurar el modelo de replicación de MySQL, tal como se muestra en la siguiente figura.

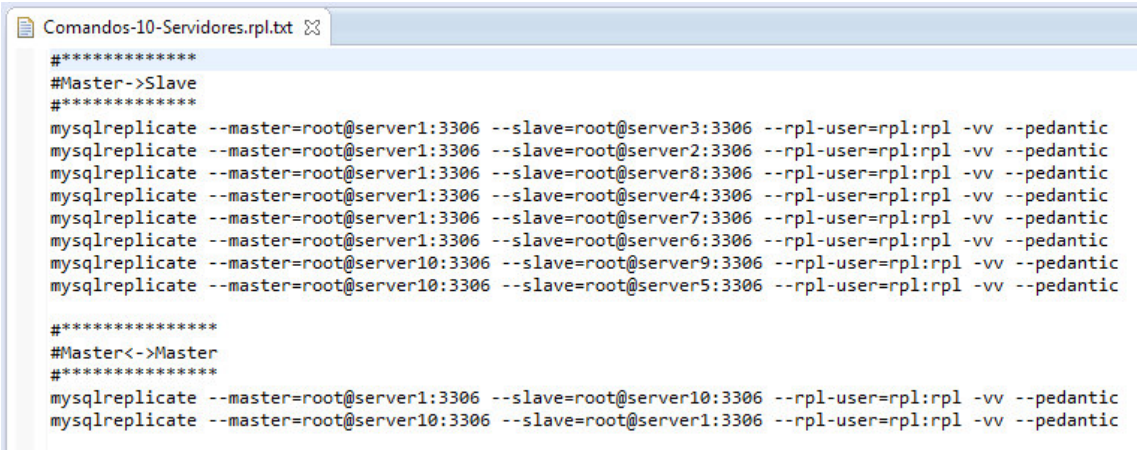


Uso de Herramientas – Generación de Comandos de un Modelo con 10 Servidores

En la siguiente figura se muestra la aplicación de la herramienta MySQL Replication Modeling para la instancia de prueba de generación de comandos a partir de un modelo de replicación con 10 servidores MySQL.

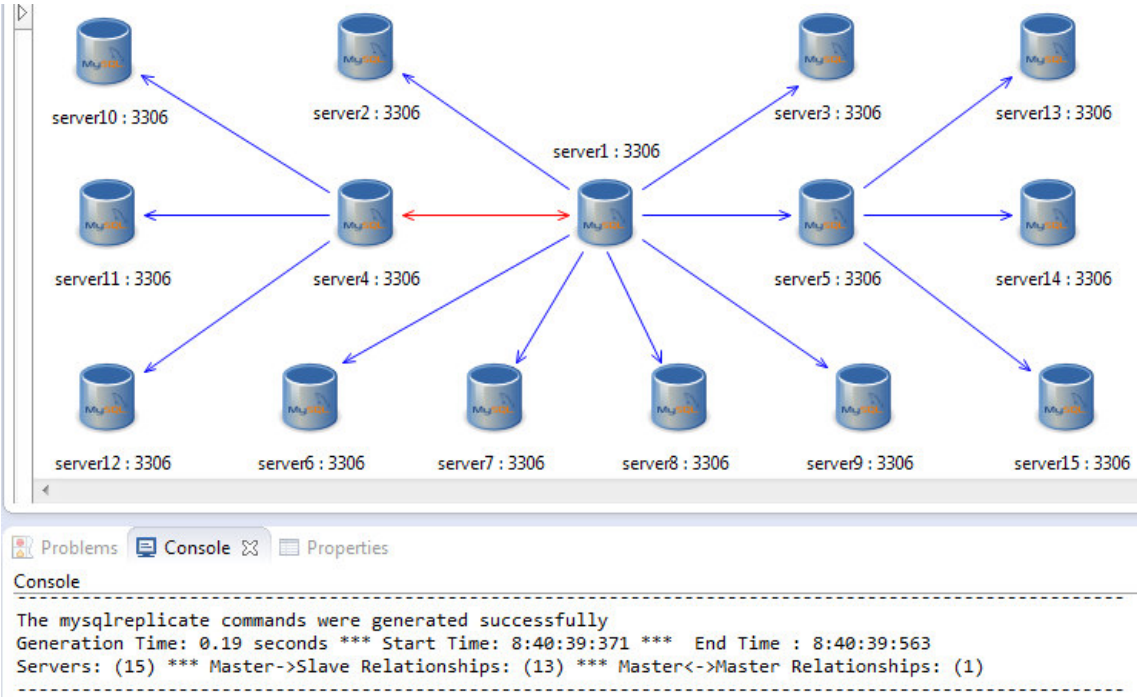


La herramienta generará un archivo de texto en el proyecto con los comandos para configurar el modelo de replicación de MySQL, tal como se muestra en la siguiente figura.

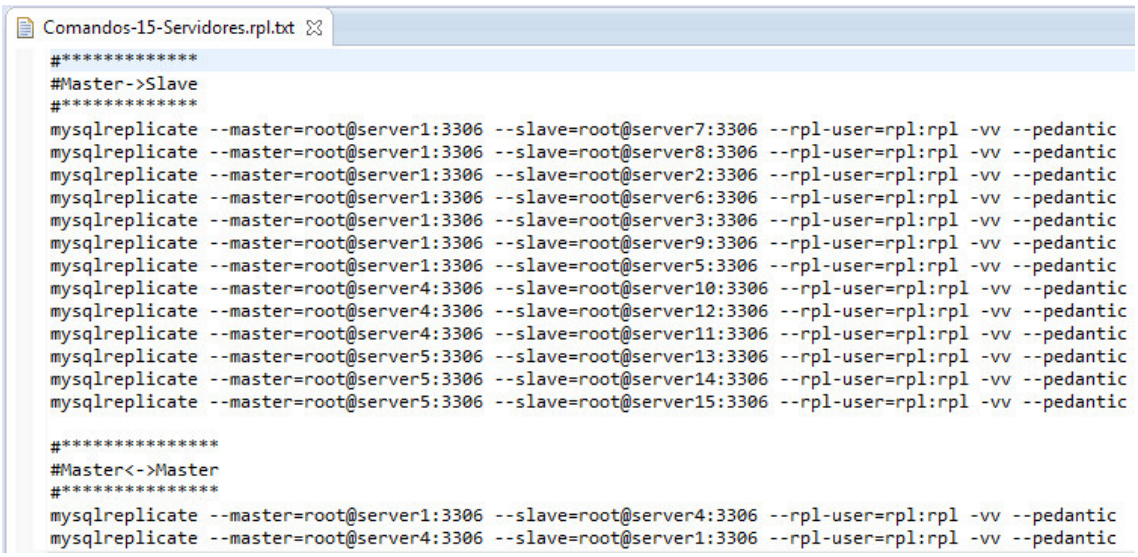


Uso de Herramientas – Generación de Comandos de un Modelo con 15 Servidores

En la siguiente figura se muestra la aplicación de la herramienta MySQL Replication Modeling para la instancia de prueba de generación de comandos a partir de un modelo de replicación con 15 servidores MySQL.

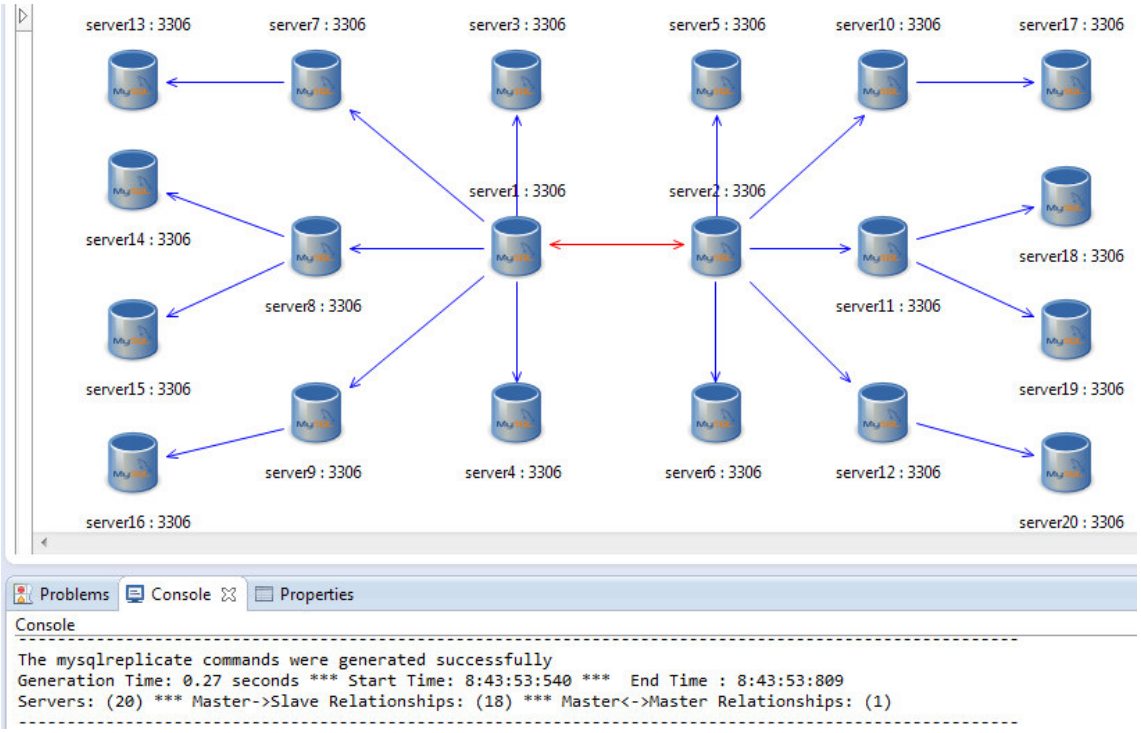


La herramienta generará un archivo de texto en el proyecto con los comandos para configurar el modelo de replicación de MySQL, tal como se muestra en la siguiente figura.

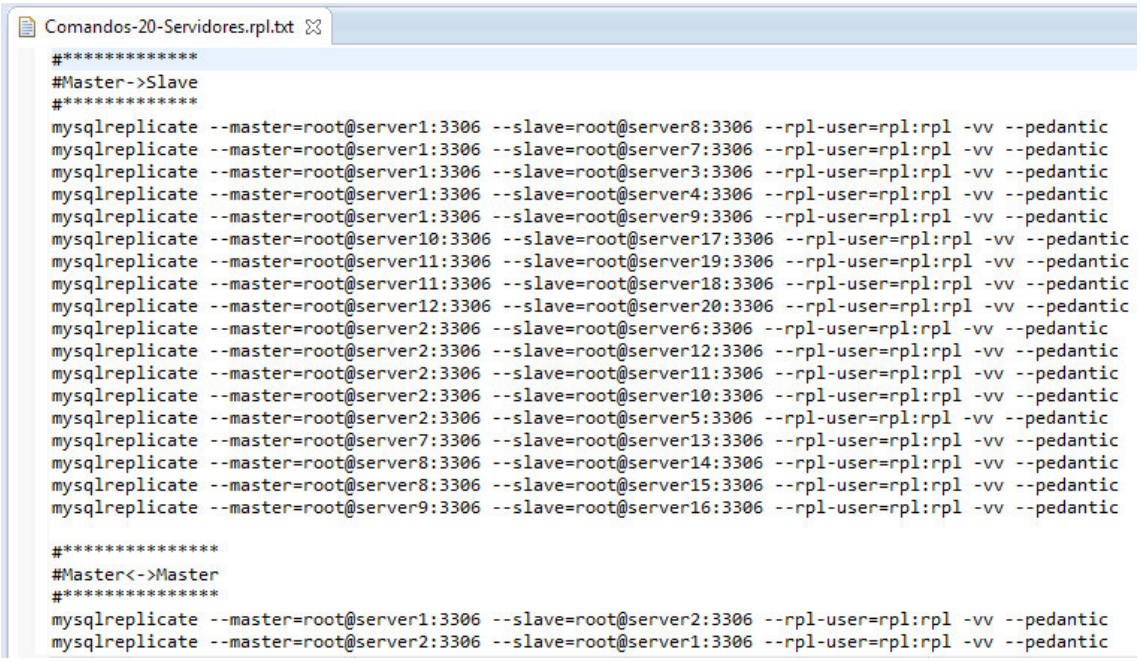


Uso de Herramientas – Generación de Comandos de un Modelo con 20 Servidores

En la siguiente figura se muestra la aplicación de la herramienta MySQL Replication Modeling para la instancia de prueba de generación de comandos a partir de un modelo de replicación con 20 servidores MySQL.

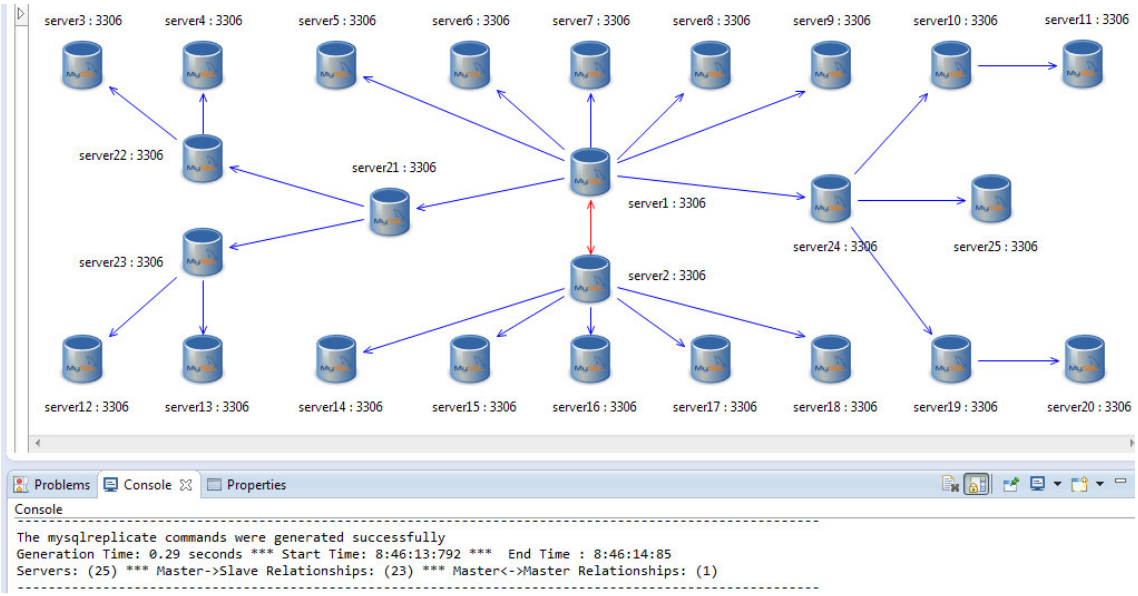


La herramienta generará un archivo de texto en el proyecto con los comandos para configurar el modelo de replicación de MySQL, tal como se muestra en la siguiente figura.

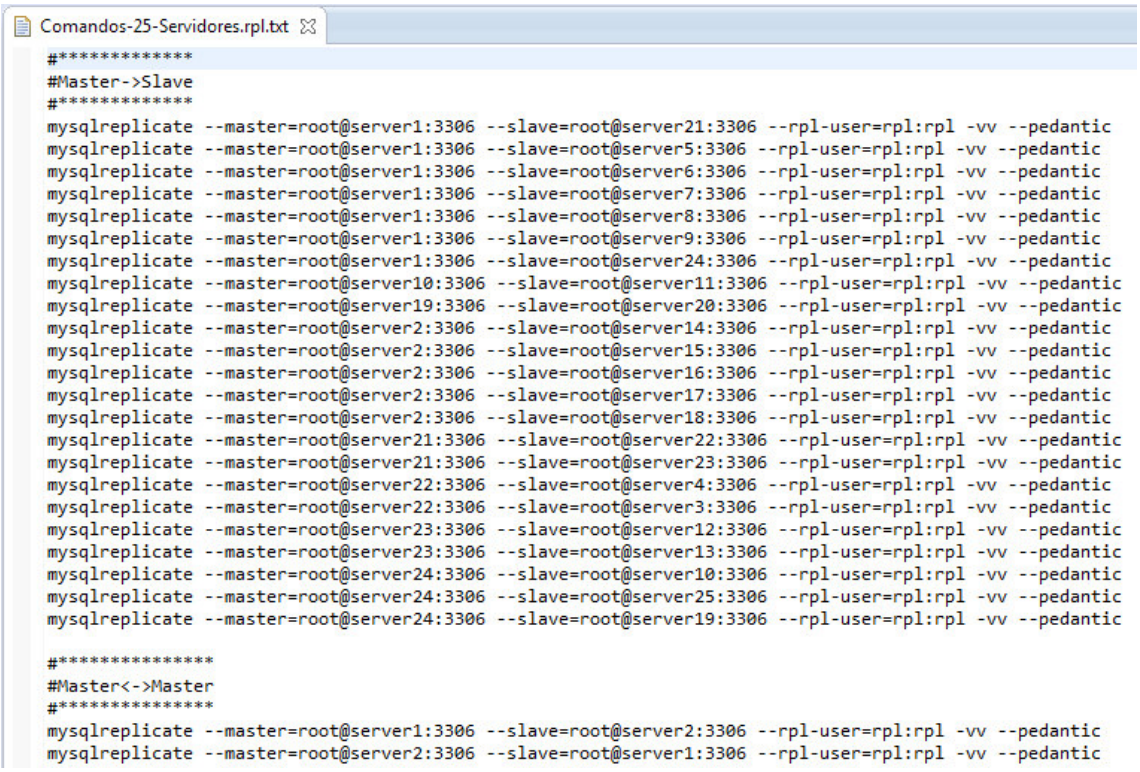


Uso de Herramientas – Generación de Comandos de un Modelo con 25 Servidores

En la siguiente figura se muestra la aplicación de la herramienta MySQL Replication Modeling para la instancia de prueba de generación de comandos a partir de un modelo de replicación con 25 servidores MySQL.



La herramienta generará un archivo de texto en el proyecto con los comandos para configurar el modelo de replicación de MySQL, tal como se muestra en la siguiente figura.



Anexo 8: Resultados de Tiempo de los Experimentos

Instancia de Prueba (Servidores del Modelo)	Tiempo (segundos)			
	Corrección de Errores		Generación de Comandos	
	Microsoft Visio 2013	MySQL Replication Modeling	Microsoft Visio 2013	MySQL Replication Modeling
5	92	33	81	0.78
	90	35	83	0.80
	95	34	79	0.78
	89	36	80	0.79
	94	33	82	0.78
	97	36	81	0.78
	91	34	83	0.79
	92	35	82	0.80
	90	35	84	0.79
	92	33	85	0.79
10	193	48	159	0.81
	195	47	161	0.81
	191	50	158	0.82
	188	51	164	0.81
	194	49	159	0.81
	192	51	162	0.81
	195	49	164	0.81
	194	48	161	0.81
	193	50	165	0.81
	195	50	162	0.81
15	328	51	239	0.83
	334	50	242	0.82
	325	52	238	0.83
	335	53	241	0.83
	329	52	244	0.84
	330	51	239	0.83
	332	50	240	0.82
	334	52	243	0.83
	333	52	242	0.83
	330	50	242	0.83

Instancia de Prueba (Servidores del Modelo)	Tiempo (segundos)			
	Corrección de Errores		Generación de Comandos	
	Microsoft Visio 2013	MySQL Replication Modeling	Microsoft Visio 2013	MySQL Replication Modeling
20	421	52	295	0.86
	418	53	298	0.86
	423	54	294	0.85
	425	55	296	0.86
	420	52	294	0.86
	422	55	290	0.86
	419	53	291	0.86
	420	55	293	0.86
	422	54	296	0.86
	421	54	295	0.86
25	595	73	398	0.87
	590	75	401	0.89
	601	74	395	0.87
	594	76	396	0.88
	591	74	398	0.88
	595	75	399	0.89
	599	76	400	0.88
	598	77	398	0.88
	596	74	395	0.87
	598	75	396	0.87